(12) UK Patent Application (19) GB (11) 2 263 002 (13) A

(43) Date of A publication 07.07.1993

(21) Application No 9227180.8

(22) Date of filing 31.12.1992

(30) Priority data
(31) 820304 (32) 06.01.1992 (33) US

(51) INT CL[5]
G06F 7/50

(52) UK CL (Edition L)
G4A AAC

(56) Documents cited
GB 2226165 A US 3993891 A

(58) Field of search
UK CL (Edition L) G4A AAC
INT CL[5] G06F 7/50
Online database: WPI

(71) Applicant
Intel Corporation

(Incorporated in the USA - Delaware)

2200 Mission College Boulevard, Santa Clara, California 95052, United States of America

(72) Inventor
Jack T Poon

(74) Agent and/or Address for Service
Potts, Kerr & Co
15 Hamilton Square, Birkenhead, Merseyside, L41 6BR, United Kingdom

(54) Parallel binary adder

(57) An N-bit binary adder with a highly parallel structure comprises a multiplicity of parallel modulo-2 adders for forming the sum of corresponding operand and carry bits. The carry inputs are generated by a conditional carry propagation generator and an unconditional carry generator from which carry bits are generated in $\log_2 2N$ operational levels.

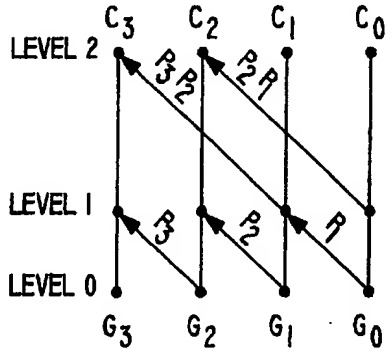


FIG. 6

GB 2 263 002 A

At least one drawing originally filed was informal and the print reproduced here is taken from a later filed formal copy.

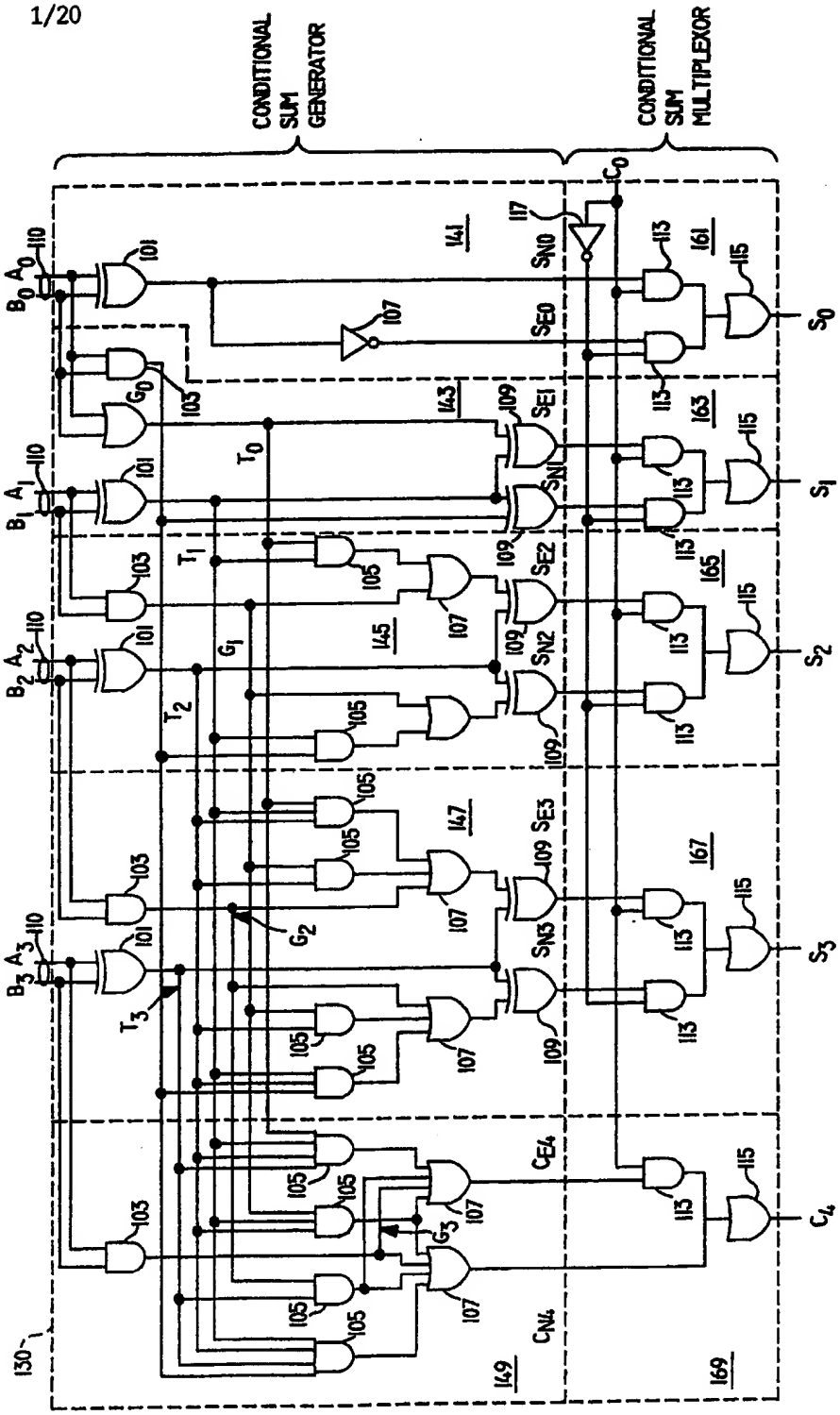


FIG. 1
(PRIOR ART)

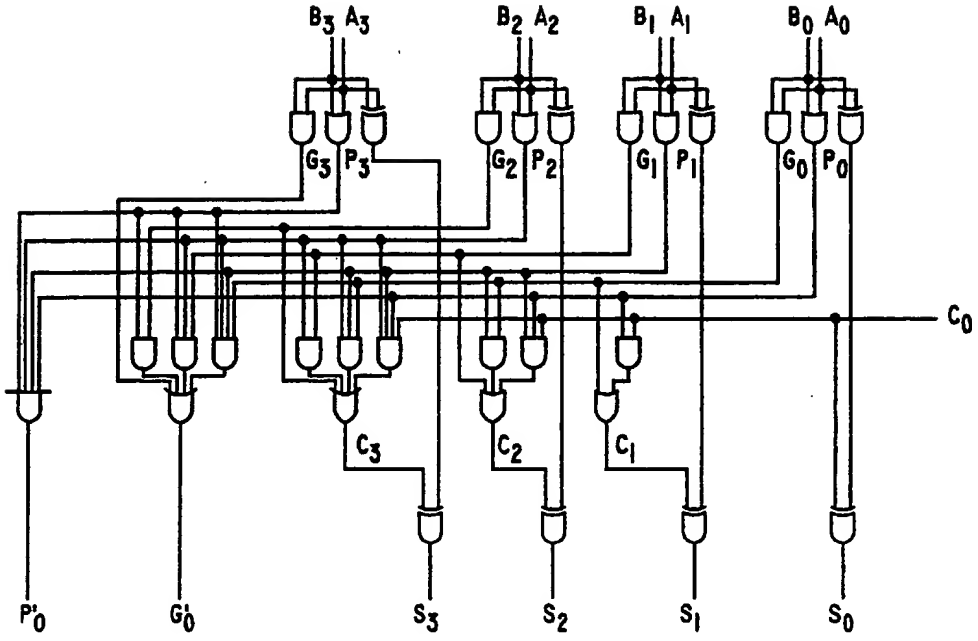


FIG. 2
(PRIOR ART)

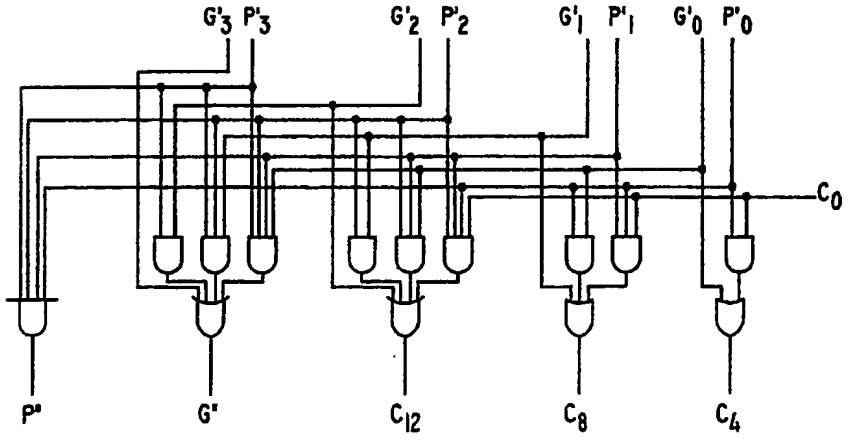


FIG. 3
(PRIOR ART)

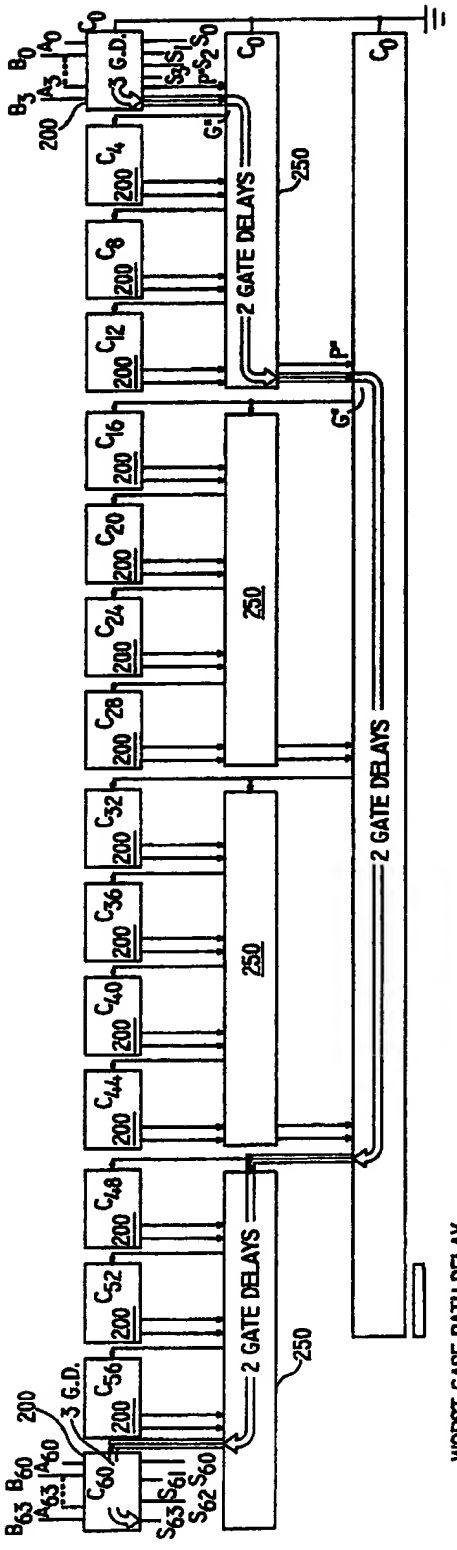


FIG. 4
(PRIOR ART)

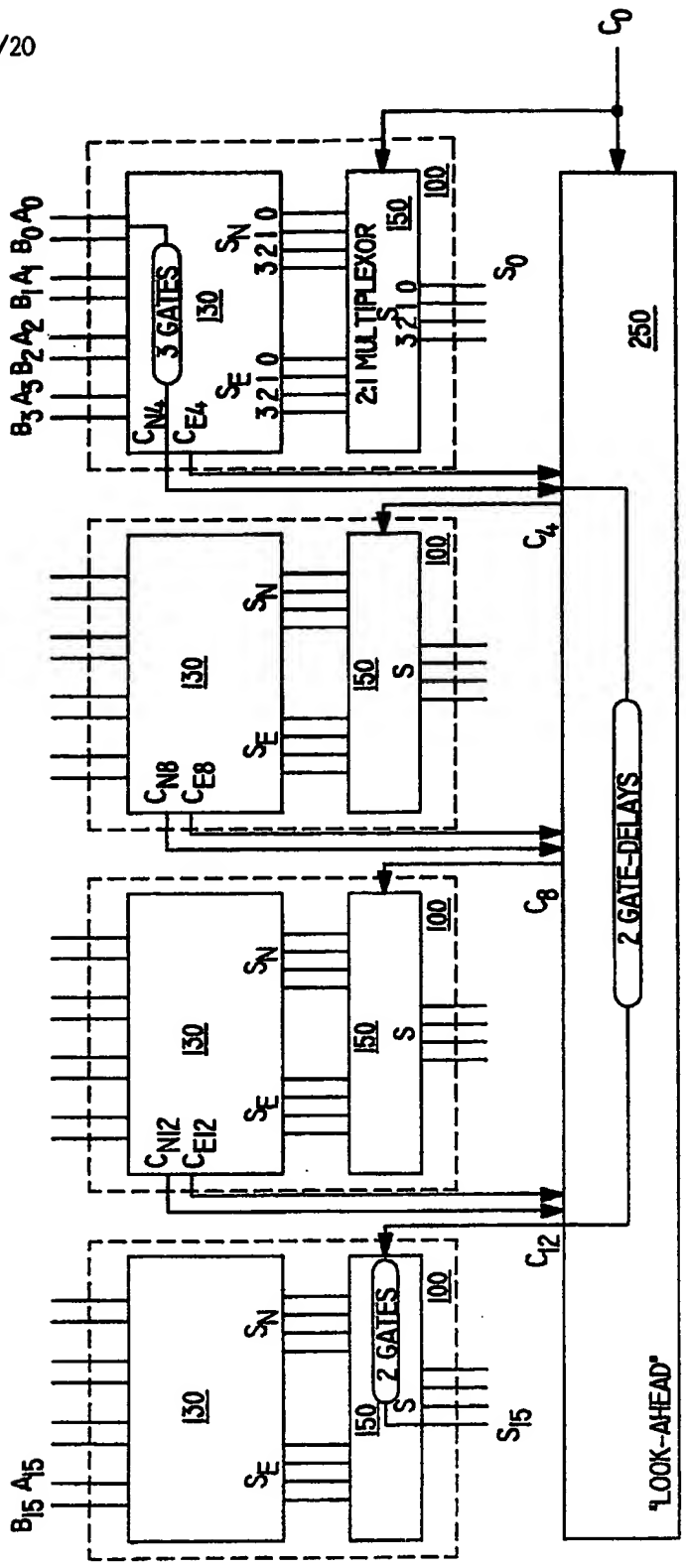


FIG. 5
(PRIOR ART)

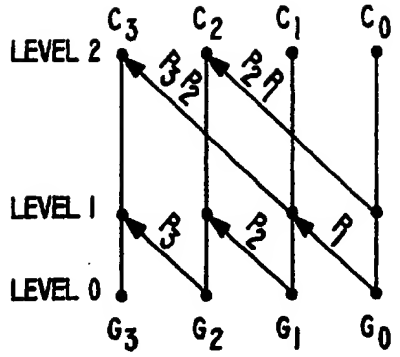

C3
C2
C1
C0
LEVEL 2
LEVEL 1
LEVEL 0
G3
G2
G1
G0

FIG. 6

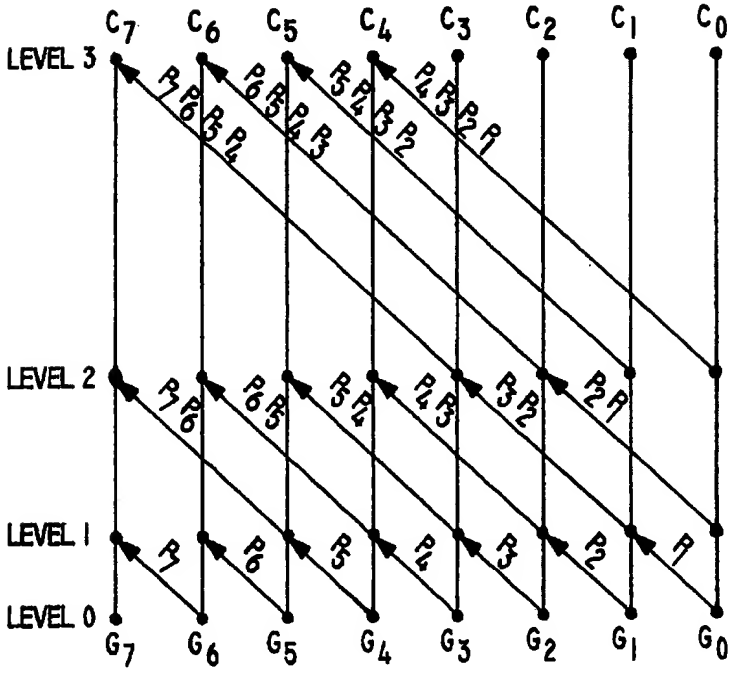

C7
C6
C5
C4
C3
C2
C1
C0
LEVEL 3
LEVEL 2
LEVEL 1
LEVEL 0
G7
G6
G5
G4
G3
G2
G1
G0

FIG. 7

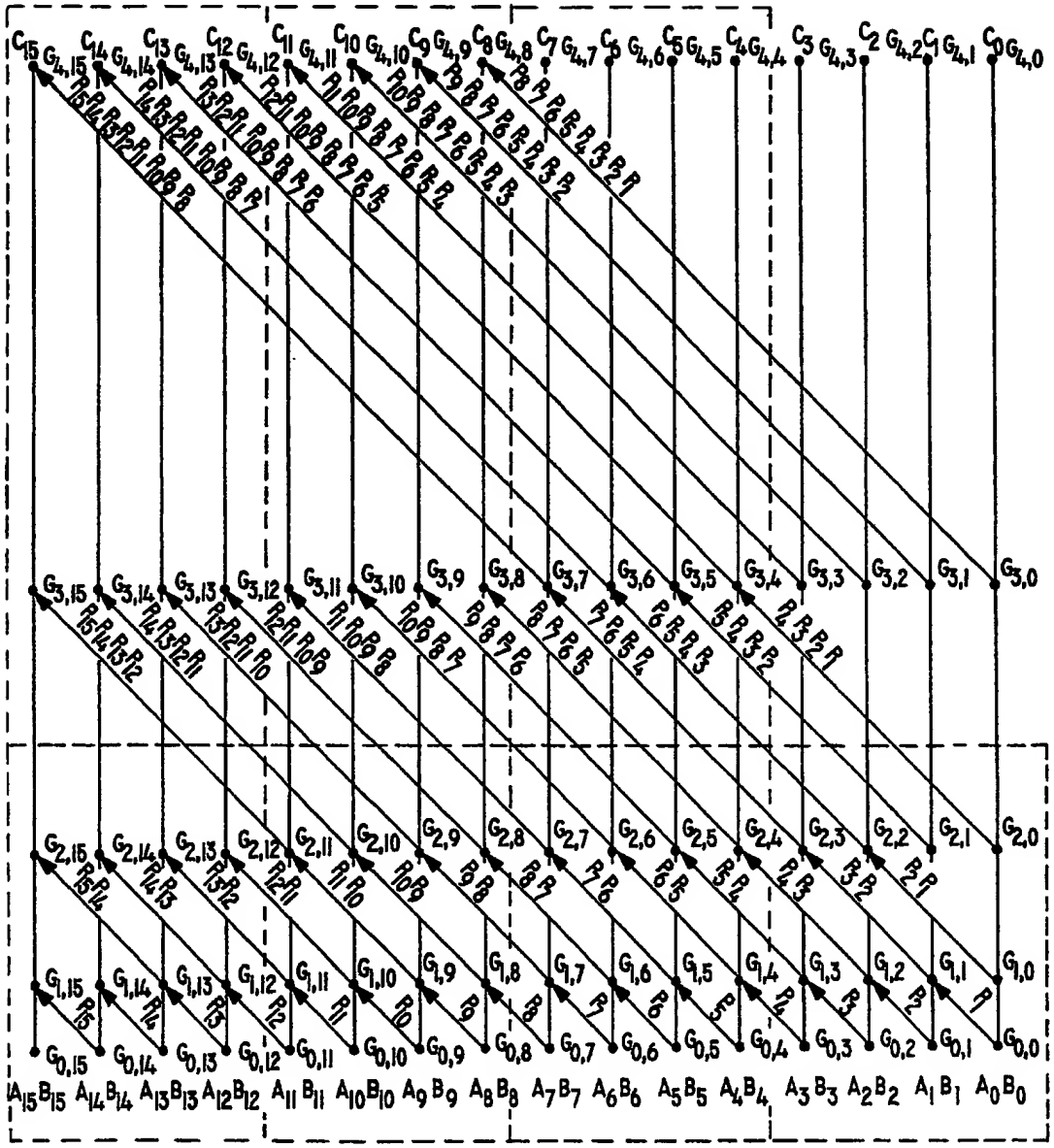

C15 G4,15 C14 G4,14 C13 G4,13 C12 G4,12 C11 G4,11 C10 G4,10 C9 G4,9 C8 G4,8 C7 G4,7 C6 G4,6 C5 G4,5 C4 G4,4 C3 G4,3 C2 G4,2 C1 G4,1 C0 G4,0
G3,15 G3,14 G3,13 G3,12 G3,11 G3,10 G3,9 G3,8 G3,7 G3,6 G3,5 G3,4 G3,3 G3,2 G3,1 G3,0
G2,15 G2,14 G2,13 G2,12 G2,11 G2,10 G2,9 G2,8 G2,7 G2,6 G2,5 G2,4 G2,3 G2,2 G2,1 G2,0
G1,15 G1,14 G1,13 G1,12 G1,11 G1,10 G1,9 G1,8 G1,7 G1,6 G1,5 G1,4 G1,3 G1,2 G1,1 G1,0
G0,15 G0,14 G0,13 G0,12 G0,11 G0,10 G0,9 G0,8 G0,7 G0,6 G0,5 G0,4 G0,3 G0,2 G0,1 G0,0
A15B15 A14B14 A13B13 A12B12 A11B11 A10B10 A9B9 A8B8 A7B7 A6B6 A5B5 A4B4 A3B3 A2B2 A1B1 A0B0

FIG. 8

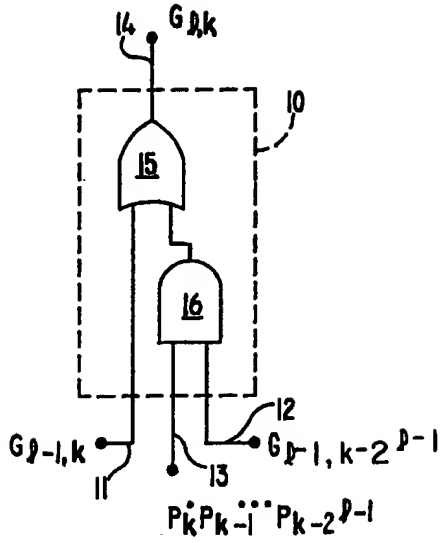

14
G j,k
10
15
16
12
G j-1, k
G j-1, k-2 j-1
11
13
Pk Pk-1 ··· Pk-2 j-1

FIG. 9

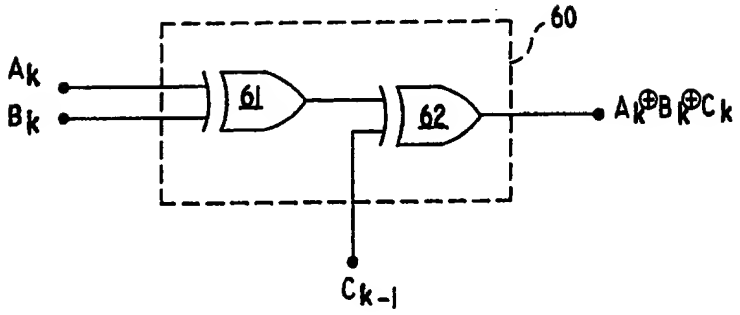
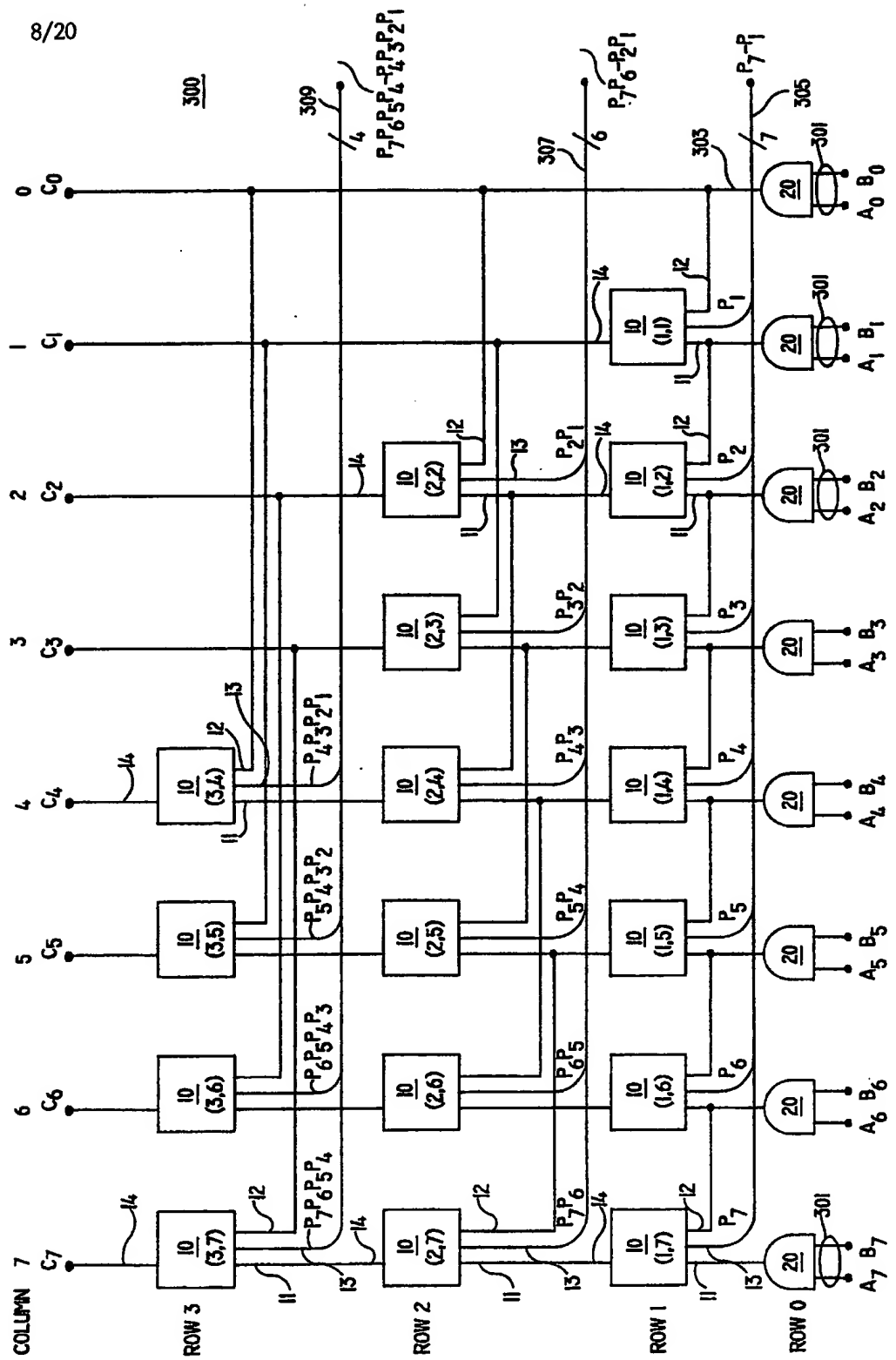

60
Ak
Bk
61
62
Ak⊕Bk⊕Ck
Ck-1

FIG. 12

FIG. 10


300
309
307
303
305
301
COLUMN 7
ROW 3
ROW 2
ROW 1
ROW 0
10 (3,7)
10 (2,7)
10 (1,7)
20

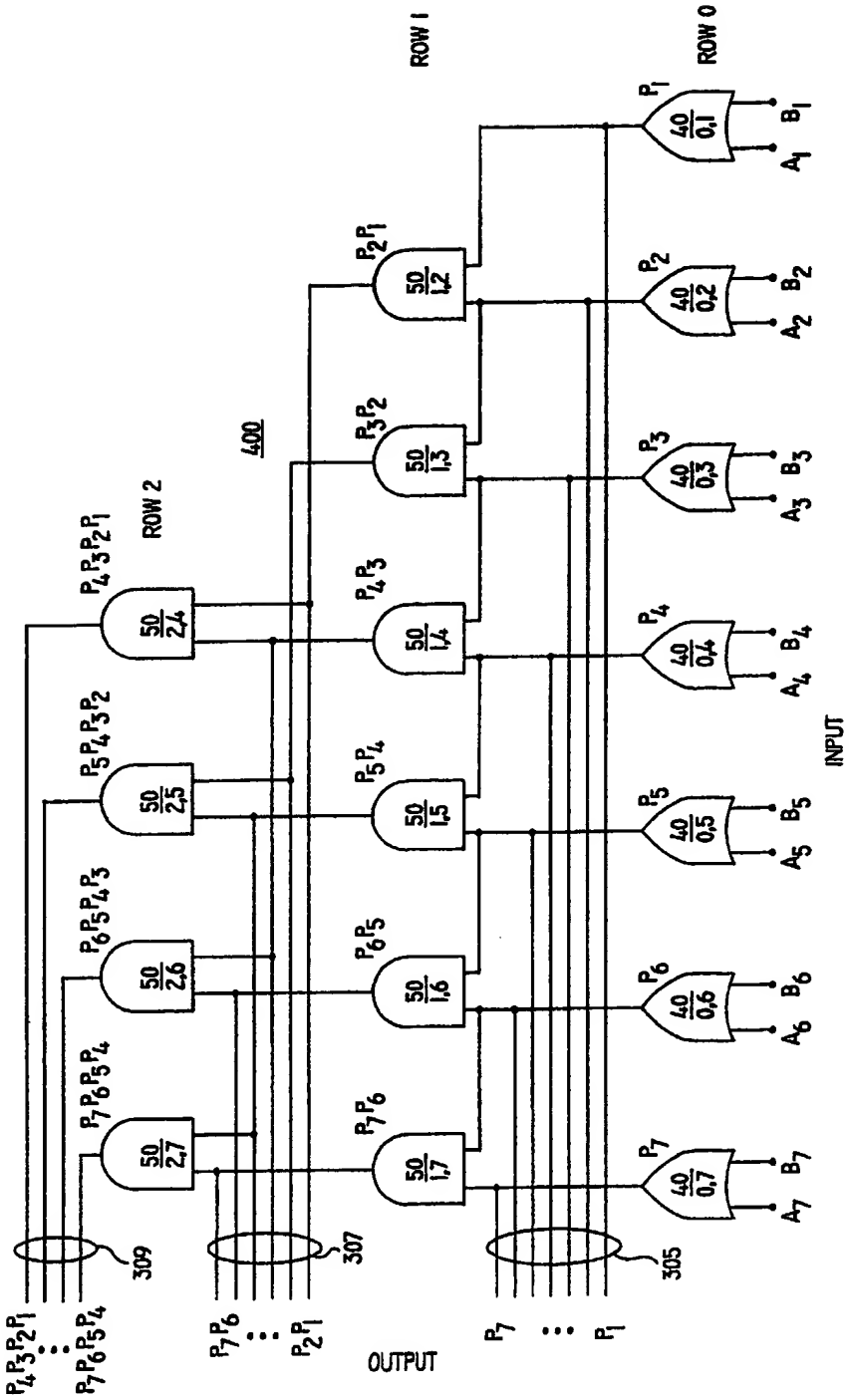

ROW 2
ROW 1
ROW 0
400
INPUT
OUTPUT
309
307
305
$P_4P_3P_2P_1$
$P_5P_4P_3P_2$
$P_6P_5P_4P_3$
$P_7P_6P_5P_4$
P_2P_1
P_3P_2
P_4P_3
P_5P_4
P_6P_5
P_7P_6
P_1
P_2
P_3
P_4
P_5
P_6
P_7
A_1 B_1
A_2 B_2
A_3 B_3
A_4 B_4
A_5 B_5
A_6 B_6
A_7 B_7
50/2,4
50/2,5
50/2,6
50/2,7
50/1,2
50/1,3
50/1,4
50/1,5
50/1,6
50/1,7
40/0,1
40/0,2
40/0,3
40/0,4
40/0,5
40/0,6
40/0,7

FIG. II

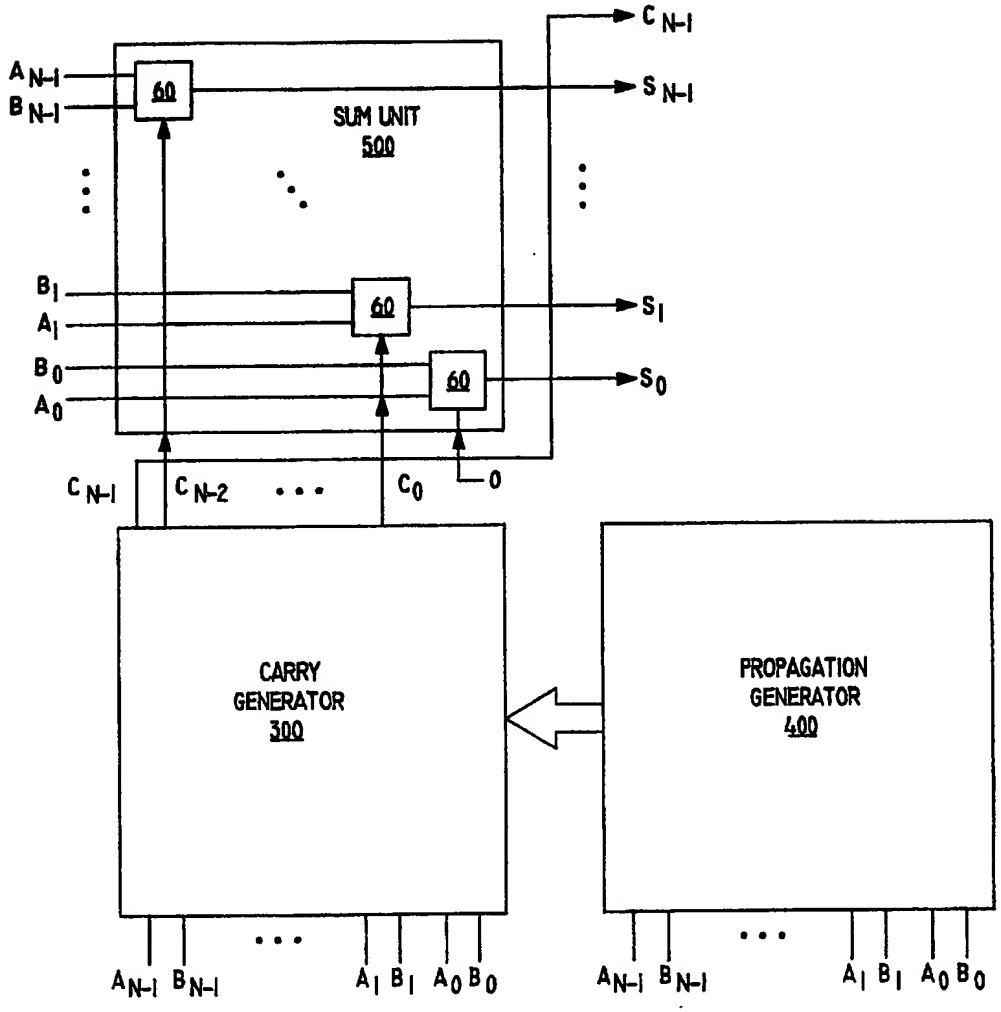

C_{N-1}
A_{N-1}
B_{N-1}
60
S_{N-1}
SUM UNIT
500
B_1
A_1
60
S_1
B_0
A_0
60
S_0
C_{N-1}
C_{N-2}
C_0
0
CARRY
GENERATOR
300
PROPAGATION
GENERATOR
400
A_{N-1} B_{N-1} A_1 B_1 A_0 B_0
A_{N-1} B_{N-1} A_1 B_1 A_0 B_0

FIG. 13

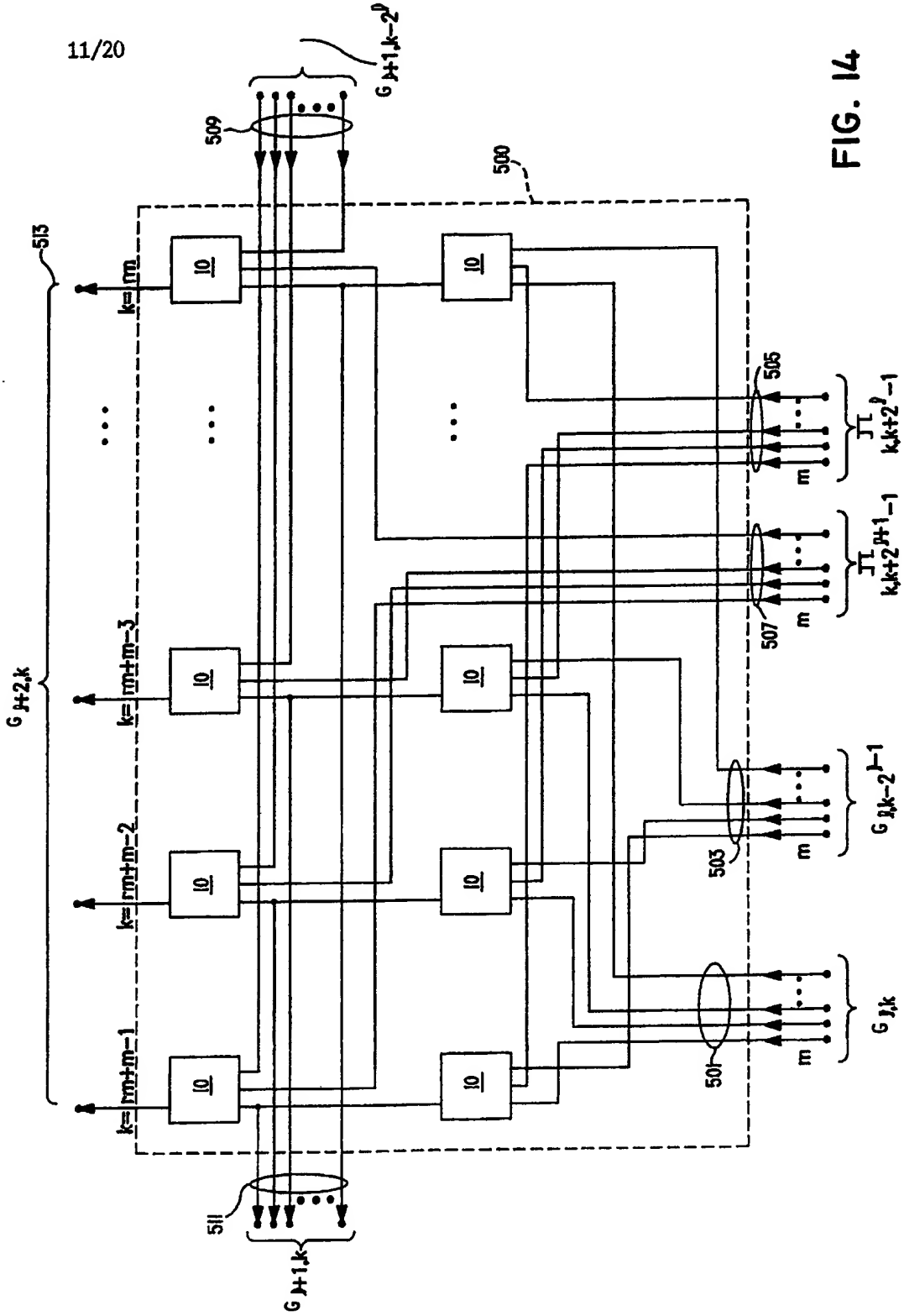
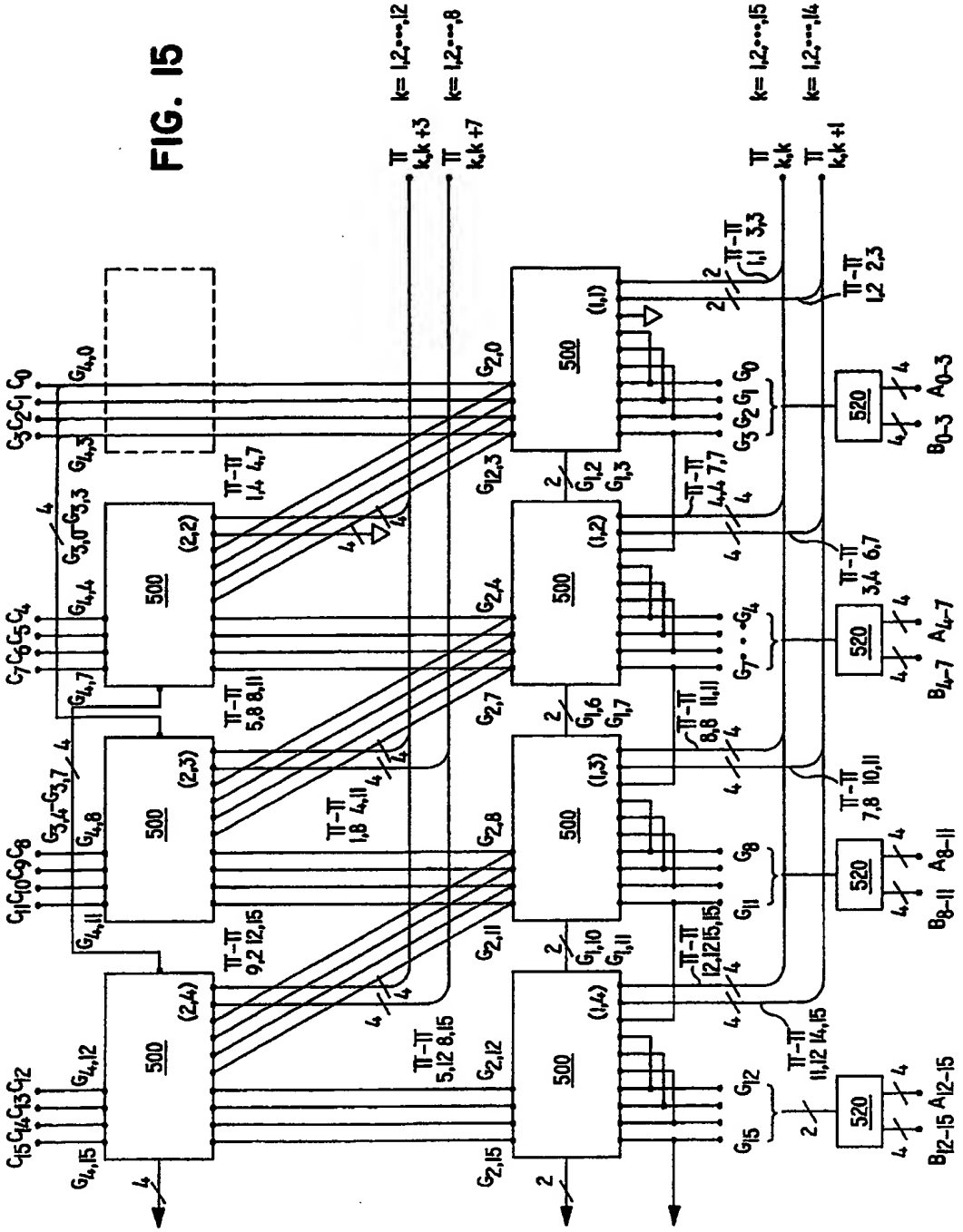

$G_{j+1,k-2^l}$
509
500
513
k=m
k=m+m-3
k=m+m-2
k=m+m-1
10
$G_{j+2,k}$
505
507
503
501
$\prod_{k,k+2^l-1}$
$\prod_{k,k+2^{j+1}-1}$
$G_{1,k-2^{j-1}}$
$G_{1,k}$
m
511
$G_{j+1,k}$

FIG. 14

FIG. 15



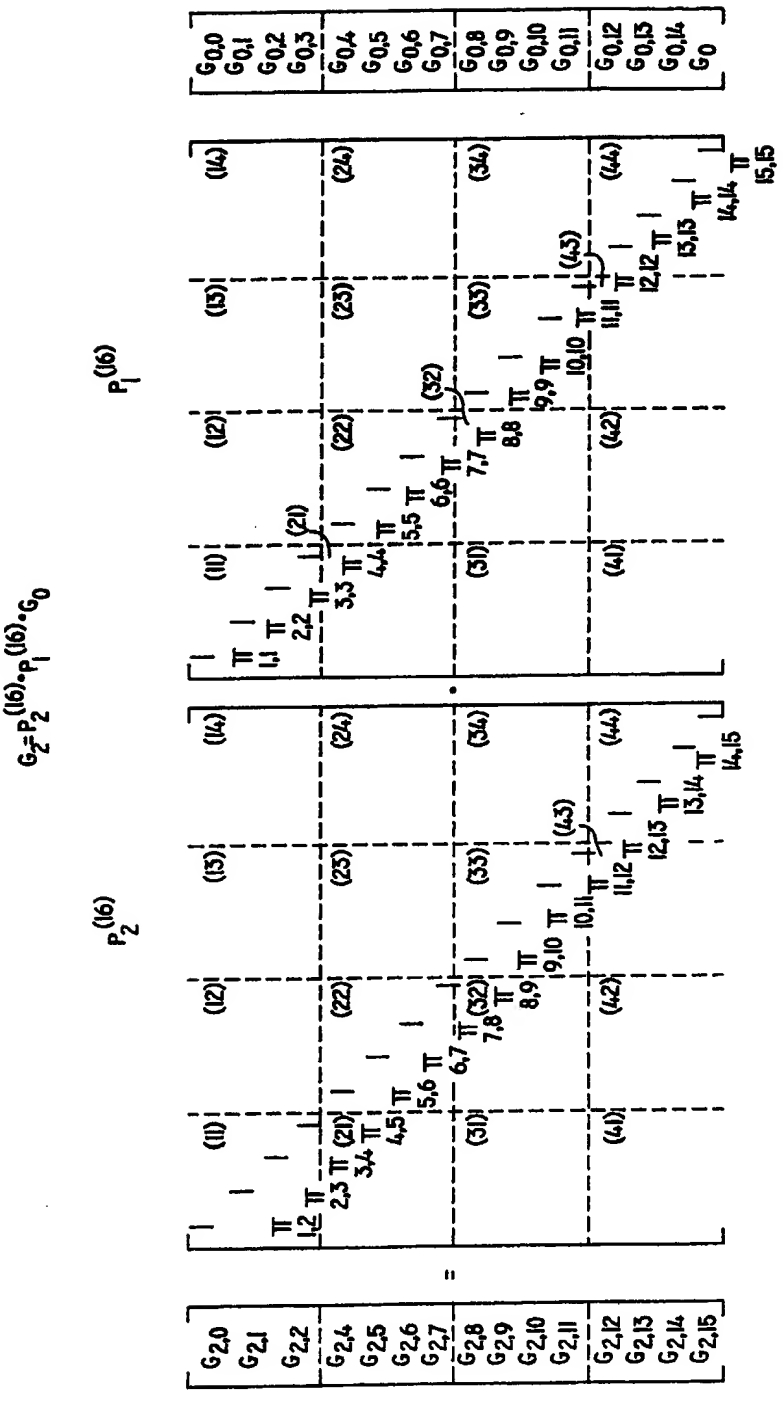


FIG. 16

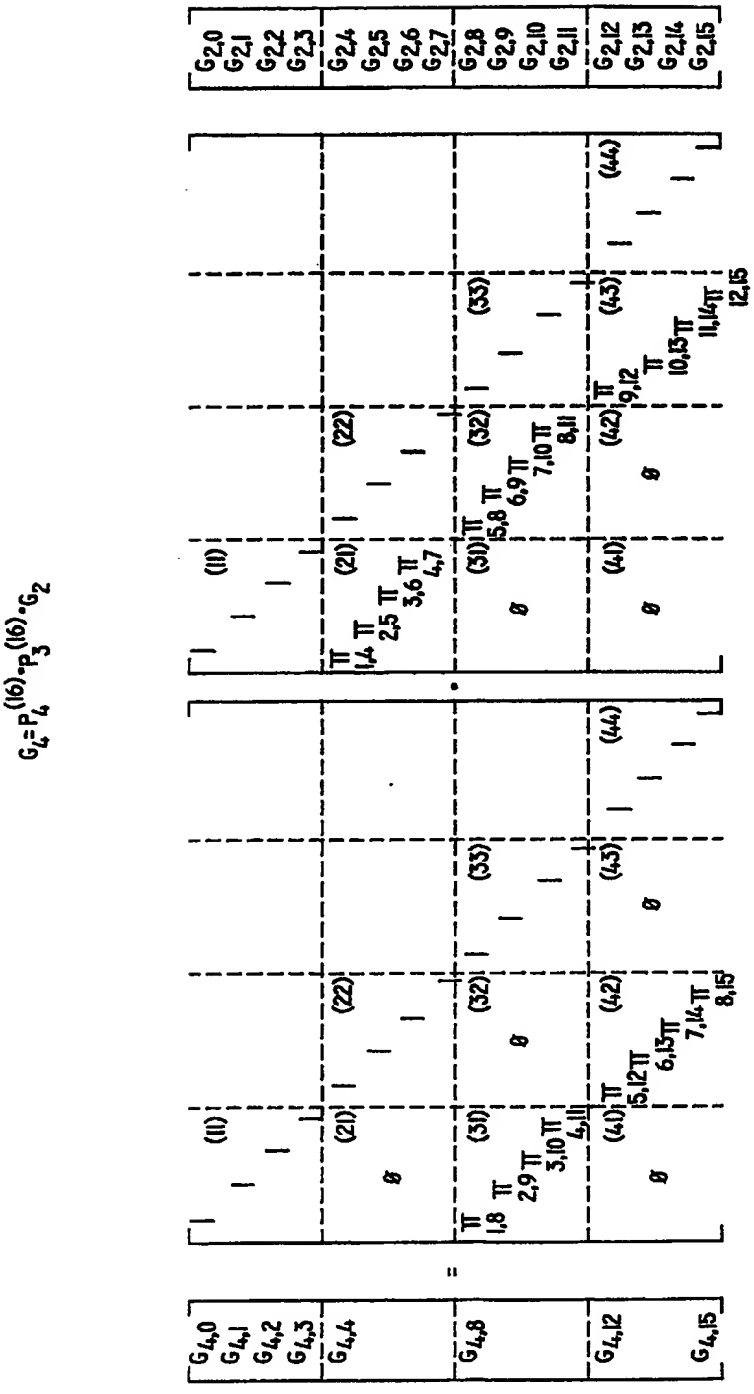
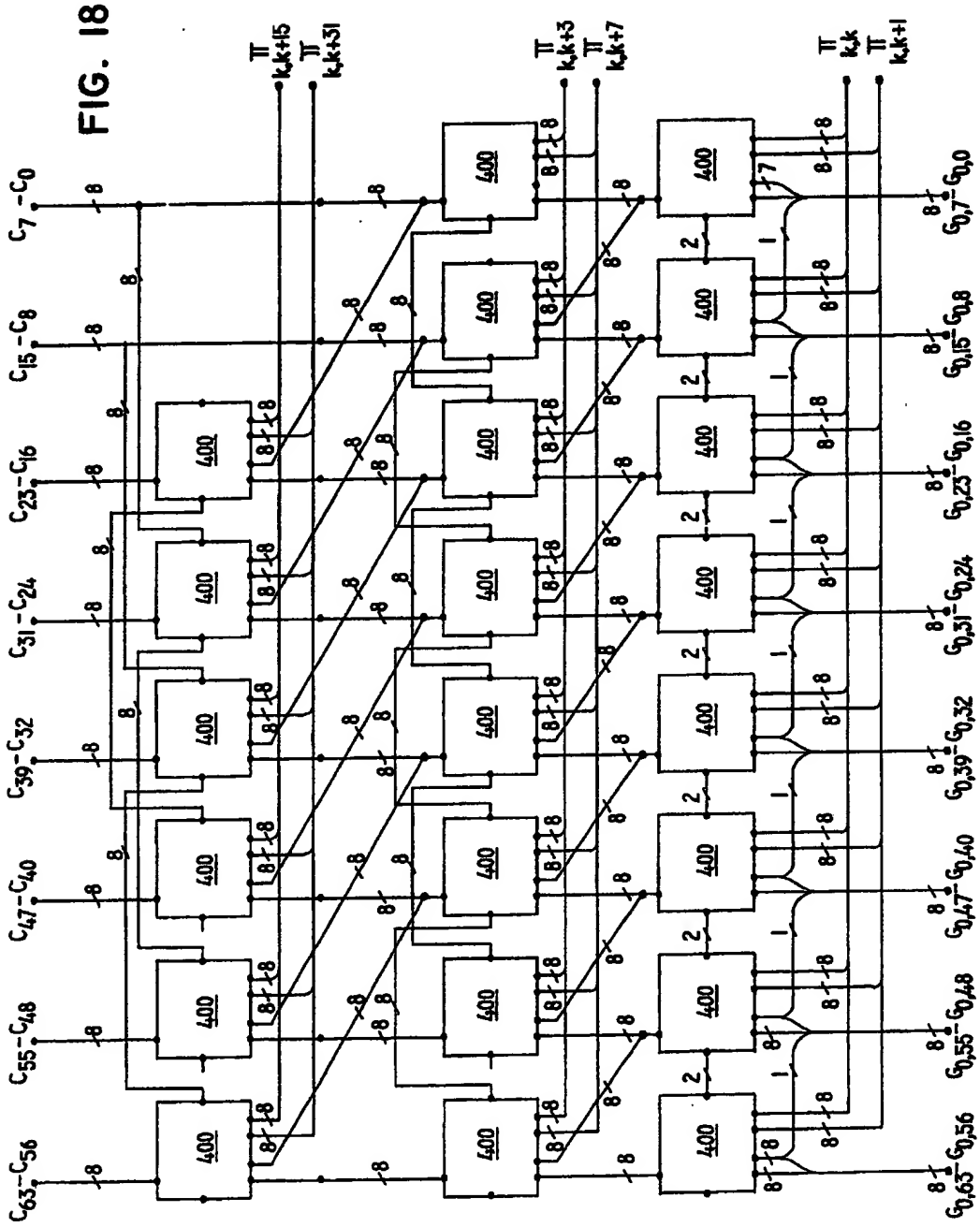


FIG. 17

FIG. 18


C7-C0
C15-C8
C23-C16
C31-C24
C39-C32
C47-C40
C55-C48
C63-C56
400
G0,7-G0,0
G0,15-G0,8
G0,23-G0,16
G0,31-G0,24
G0,39-G0,32
G0,47-G0,40
G0,55-G0,48
G0,63-G0,56

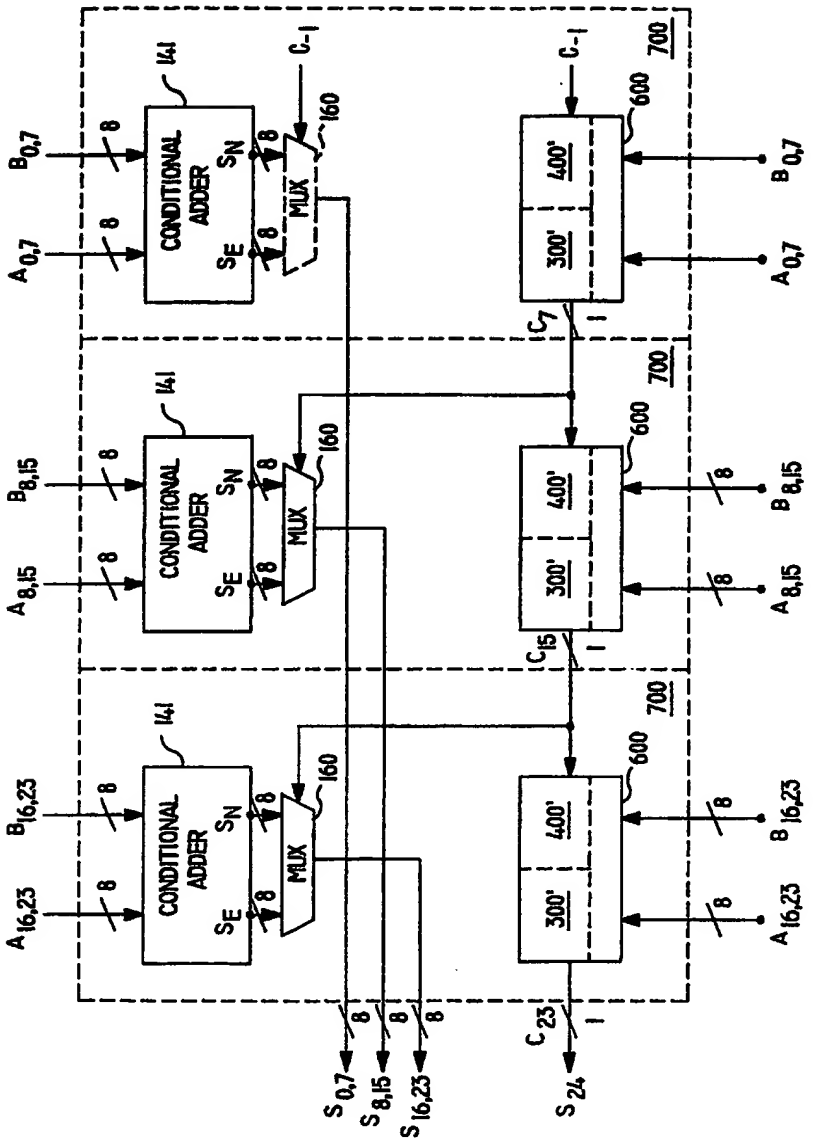

CONDITIONAL ADDER
MUX
141
160
600
700
400'
300'
C-1
C7
C15
C23
B0,7
A0,7
B8,15
A8,15
B16,23
A16,23
S0,7
S8,15
S16,23
S24
8
1

FIG. 19

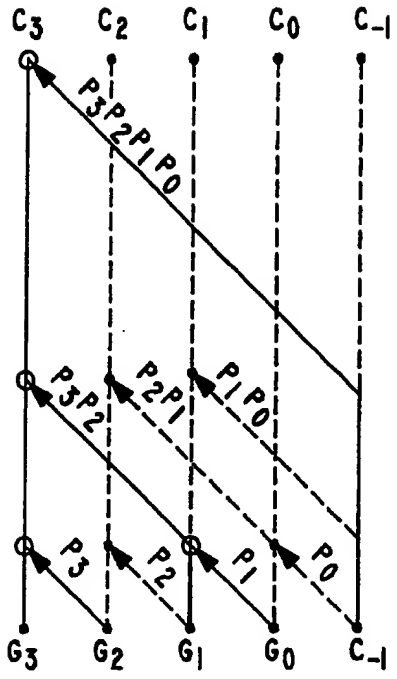

C3
C2
C1
C0
C-1
P3P2P1P0
P3P2
P2P1
P1P0
P3
P2
P1
P0
G3
G2
G1
G0
C-1

FIG. 20

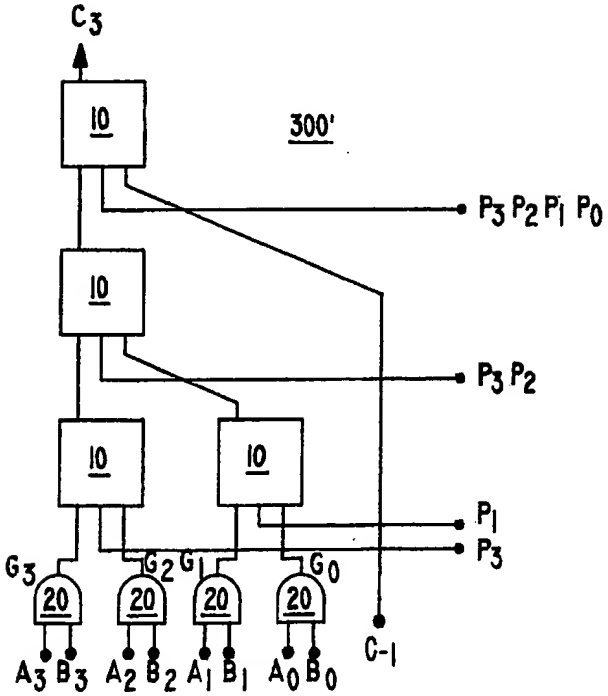

C3
10
300'
P3 P2 P1 P0
10
P3 P2
10
10
P1
P3
G3
G2
G1
G0
20
20
20
20
C-1
A3 B3
A2 B2
A1 B1
A0 B0

FIG. 21

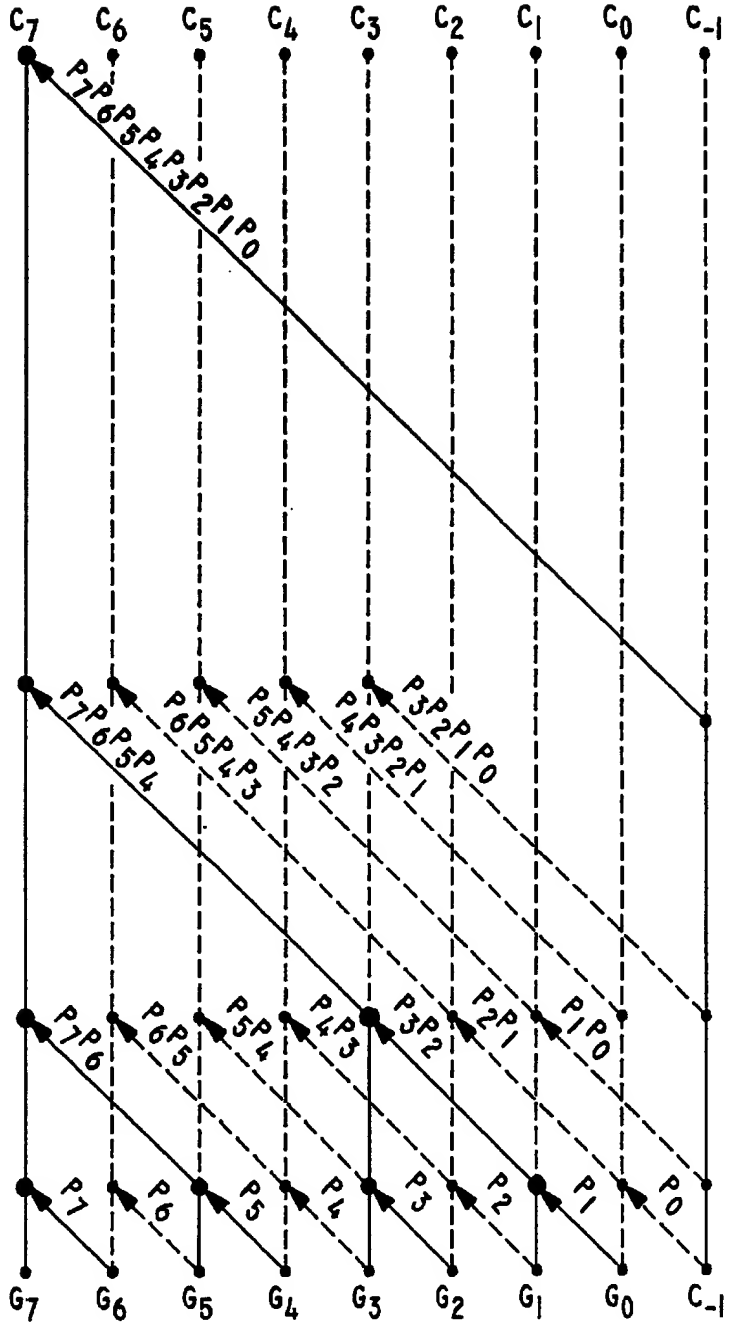

C7 C6 C5 C4 C3 C2 C1 C0 C-1
P7P6P5P4P3P2P1P0
P6P5P4P3P2P1P0
P5P4P3P2P1P0
P4P3P2P1P0
P3P2P1P0
P7P6P5P4
P6P5P4P3
P5P4P3P2
P4P3P2P1
P3P2P1P0
P7P6
P6P5
P5P4
P4P3
P3P2
P2P1
P1P0
P7
P6
P5
P4
P3
P2
P1
P0
G7 G6 G5 G4 G3 G2 G1 G0 C-1

FIG. 22

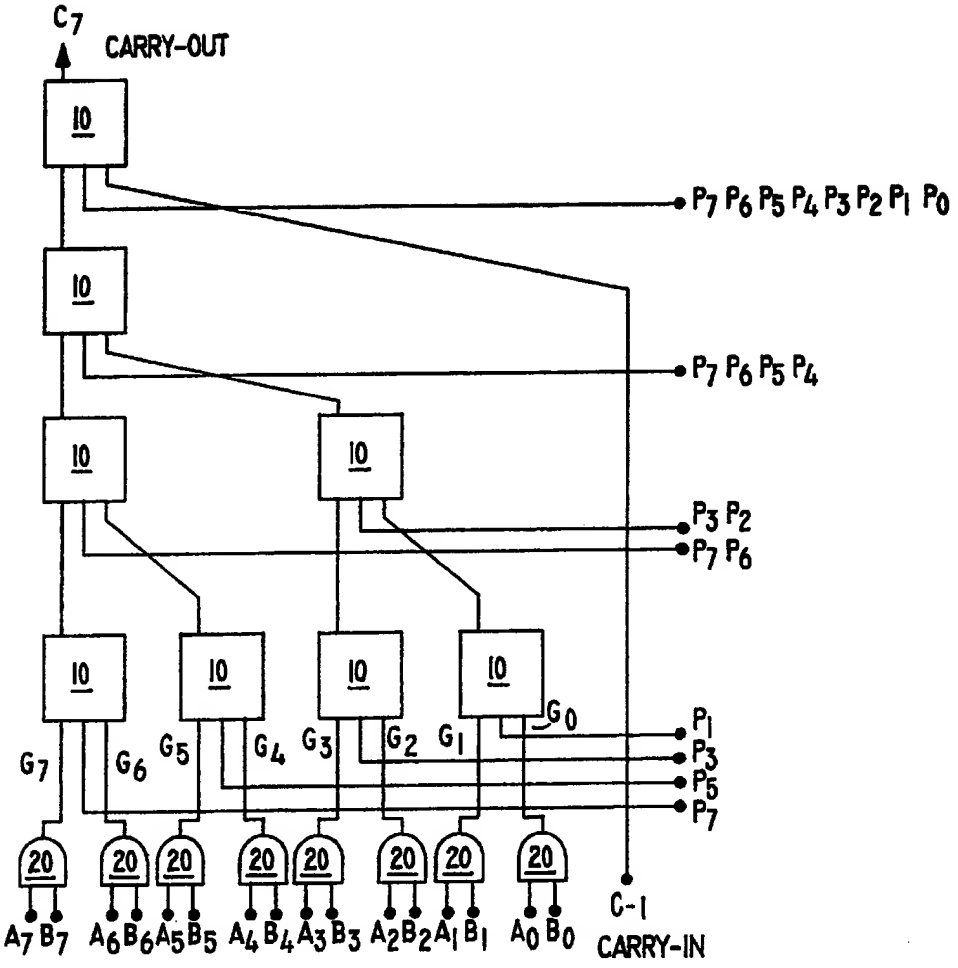
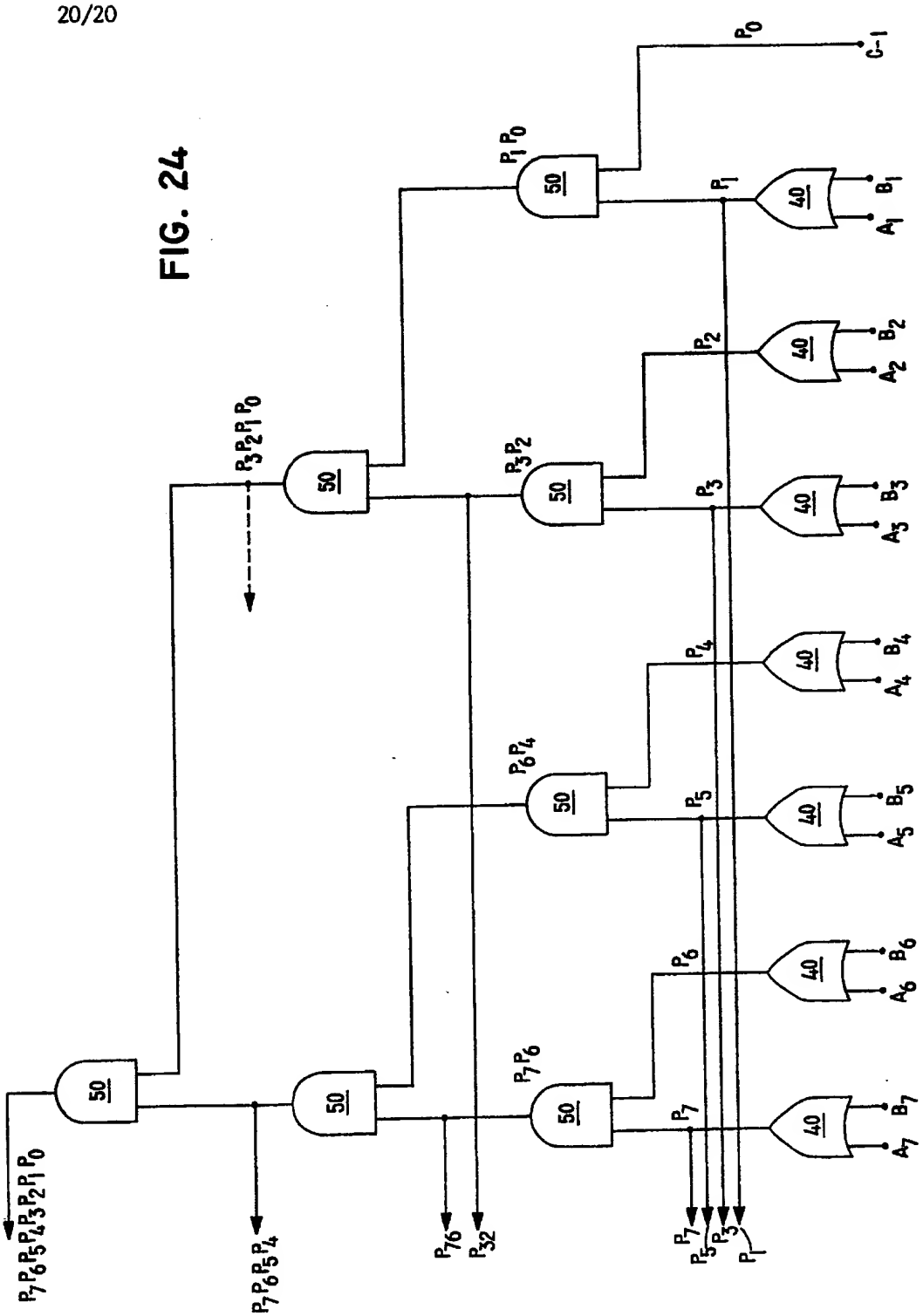

C7
CARRY-OUT
10
10
10
10
10
10
10
10
P7 P6 P5 P4 P3 P2 P1 P0
P7 P6 P5 P4
P3 P2
P7 P6
G0
P1
P3
P5
P7
G7
G6
G5
G4
G3
G2
G1
20
20
20
20
20
20
20
20
C-1
CARRY-IN
A7 B7
A6 B6
A5 B5
A4 B4
A3 B3
A2 B2
A1 B1
A0 B0

FIG. 23

FIG. 24


P0
C-1
P1P0
50
P1
40
B1
A1
P2
40
B2
A2
P3P2P1P0
50
P3P2
50
P3
40
B3
A3
P4
40
B4
A4
P6P4
50
P5
40
B5
A5
P6
40
B6
A6
P7P6
50
P7
40
B7
A7
P7P6P5P4P3P2P1P0
P7P6P5P4
P76
P32
P7
P5
P3
P1

A PARALLEL BINARY ADDER

FIELD OF INVENTION

The invention pertains to the field of arithmetic adder circuits and more specifically to binary adder networks.

BACKGROUND TO THE INVENTION

Binary adder networks are basic to digital computer arithmetic operation. Because of the large number of adder operations involved, the history of computer development shows a constant search for faster adder networks either through faster component technology or by improved network organization using various auxiliary logic or computing networks to augment the basic adder unit.

Early digital computers used ripple-carry adders in which the i^{th} adder output bit may be represented by the modulo-2 bit sum

$$S_i = A_i \oplus B_i \oplus C_{i-1}$$

where A_i and B_i are the i^{th} bit of the input operands, and C_{i-1} is the carry-in from the next lowest bit sum. The carry-in may be represented in terms of the prior stage operands (A_{i-1}, B_{i-1}) and the prior stage carry-in, C_{i-2}, as $C_{i-1} = A_{i-1} \cdot B_{i-1} + C_{i-2}(A_{i-1} + B_{i-1})$ where ($\cdot$, +) are Boolean (AND, OR) operators respectively. The time for the carry-bits to ripple through became the limiting factor in the speed of adders. Subsequent fixed-time adders were introduced to overcome these deficiencies. These fixed-time adders may be classified into two categories: conditional sum and carry-look-ahead (CLA) adders.

Conditional adders compute each bit sum, S_i, twice: one sum, S_{NI}, based on the assumption that the carry-in bit, C_i, is zero; a second sum, S_{EI}, on the assumption that $C_i = 1$. Figure 1 is the logic diagram of a 4-bit-slice conditional sum adder. (Ref. "Introduction to Arithmetic", Waser and Flynn, Holt, Rinehart and Winston, 1982, p. 77ff). The two input operands are represented by input bits A_0, A_1, A_2, A_3 and B_0, B_1, B_2, B_3, respectively. Each pair of operand bits (A_i, B_i) are applied to input terminals 110. A_0, B_0 correspond to the input operand least significant bit while A_3, B_3 correspond to the most significant bits. The conditional sum adder consists of two basic

sections: the conditional sum generator unit 130 that forms at its output the two sets of conditional sums and conditional carry, S_{N0}, S_{N1}, S_{N2},S_{N3}, C_{N4} and S_{E0}, S_{E1}, S_{E2}, S_{E3}, C_{E4}, the latter group being based on the assumption of a non-zero carry-in to its corresponding individual conditional sum generator 141, 143, 145, 147, 149, respectively. These conditional signals are applied to conditional sum selector unit 150 consisting of the individual output selectors 161, 163, 165, 167, 169 corresponding to output sum bits S_0, S_1, S_2, S_3 and output carry bit C4. The selection is controlled by the carry-in bit, C_0, and its complement, $\overline{C_0}$, operating on the conditional sums by means of AND-gates 113 and OR-gates 115.

The logic equations governing the behavior of the conditional 4-bit slice adder of Figure 1 are as follows:

$$
\begin{aligned}
S_{N0} &= A_0 \oplus B_0 \\
S_{E0} &= \overline{S}_{N0} \\
S_{N1} &= A_1 \oplus B_1 \oplus G_0 \\
S_{E1} &= A_1 \oplus B_1 \oplus P_0 \\
S_{N2} &= A_2 \oplus B_2 \oplus (G_1 + T_1 G_0) \\
S_{E2} &= A_2 \oplus B_2 \oplus (G_1 + T_1 P_0) \\
S_{N3} &= A_3 \oplus B_3 \oplus (G_2 + T_2 G_1 + T_2 T_1 G_0) \\
S_{E3} &= A_3 \oplus B_3 \oplus (G_2 + T_2 G_1 + T_2 T_1 P_0) \\
C_{N4} &= G_3 + T_3 G_2 + T_3 T_2 G_1 + T_3 T_2 T_1 G_0 \\
C_{E4} &= G_3 + T_3 G_2 + T_3 T_2 G_1 + T_3 T_2 T_1 P_0
\end{aligned}
$$

where

$$
\begin{aligned}
G_I &= A_I B_I, \\
P_I &= A_I B_I, \\
T_I &= A_I \oplus B_I.
\end{aligned}
$$

The true 4-bit sum and carry-out is selected by selector unit 150 in accordance in accordance with the following boolean equations:

$$
\begin{aligned}
S_0 &= S_{E0} C_0 + S_{N0} \overline{C_0} \\
S_1 &= S_{E1} C_0 + S_{N1} \overline{C_0} \\
S_2 &= S_{E2} C_0 + S_{N2} \overline{C_0} \\
S_3 &= S_{E3} C_0 + S_{N3} \overline{C_0} \\
C_4 &= C_{E4} C_0 + C_{N4}
\end{aligned}
$$

The above concept could be extended to additional bits with the attendant increase in complexity implied by the above equations and by Figure 1.

Carry-looks ahead (CLA) adders have been the most popular integrated circuit implementation in the recent past because of their simplicity and modularity. Modularity implies relative ease in extending the number of bits in each operand by the use of identical parallel units.

Consider, for example, the 4-bit slice CLA of Figure 2. Comparison with Figure 1, a 4-bit slice conditional adder, clearly shows the relative simplicity of the CLA.

The CLA sum may be expressed in the following logic expression as

$$S_i = A_i \oplus B_i \oplus C_{i-1} \quad , \quad i = 0, 1, 2, 3$$

and the CLA carry as

$$C_i + A_iB_i + C_i (A_i + B_i)$$

or $C_i = G_i + P_iC_i$

where $G_i = A_iB_i$

and $P_i = A_i + B_i$

The above CLA sum expression can be immediately evaluated, absent the carry term (C_{i-1}), by forming the EOR of the two operands (A_i, B_i). The carry term, C_{i-1}, is a function of lower order indexed operands, (A_{i-1}, B_{i-1}), and lower order carries, C_{i-2}. As a result, the time to complete an addition is generally governed by availability of the carry-in bit to each sum-bit.

The above expression for C_i is a recursive equation, i.e., one in which the current value, C_{i+1}, is a function of its own past values. It may be explicitly stated as follows:

$$C_{i+1} = G_i + P_iG_{i-1} + P_iP_{i-1}G_{i-2} + \ldots + P_iP_{i-1} \ldots P_0C_0$$

Hence, for the four-bit case of Figure 2, the major output carry, C4, may be expressed as

$$C_4 = G_3 + P_3G_2 + P_3P_2G_1 + P_3P_2P_1G_0 + P_3P_2P_1P_0C_0$$

By substituting the following into the above expression

$$G_0' = G_3 + P_3G_2 + P_3P_2G_1 + P_3P_2P_1G_0$$

and $P_0' = P_3P_2P_1P_0C_0$ obtains $C_4 = G_0' + P_0'C_0$ which represents the logical expression for the G_0', P_0' output terminals of Figure 2.

If two networks of the type shown in Figure 2 were to be used as a modular units for generating an 8-bit sum, the carry-in bit to the higher order 4-bit network, C4, would have to be formed in accordance with the above expression. The output carry of the higher order unit, C8, would then be expressible as

$$C_8 = G_1' + P_1'G_0' + P_1'P_0'C_0$$

where G_1' and P_1' are the CLA output pair of the next higher order CLA modular unit.

Modularity was extended by means of a four group CLA generator that accommodated four CLA 4-bit slice adders and produced at output the necessary carry information, i.e., C_4, C_8, C_{12} and P", G", to form a 16-bit CLA adder using four modular adder units of the type shown in Figure 2. Figure 3 shows a four group CLA generator with four input pairs, (G_0', P_0'), (G_1', P_1'), (G_2', P_2') and (G_3', P_3') and carry outputs corresponding to C_4, C_8, C_{12} and (P", G"), where

$$G_{12} = G_2' + P_2'G_1' + P_2'P_1'G_0' + P_2'P_1'P_0'C_0$$

and

$$G'' = G_3' + P_3'G_2' + P_3'P_2'P_1'C_0$$
$$P'' = P_3'P_2'P_1'P_0'$$

Thus, the most significant carry-out bit, C16, could be logically formed as

$$C_{16} = G'' + P''C_0$$

and passed on, as needed, to higher order modular CLA adder units.

Figure 4 shows the logical extension of modular CLA concept to 64-bit addition. A total of sixteen modular 4-bit slice SLA adders 200 are arrayed in parallel to accept input operand pairs, (A_0, B_0) ... (A_3, B_3), (A_4, B_4) ... (A_7, B_7), ..., (A_{60}, B_{60}) ... (A_{63}, B_{63}) and carry-in bits, $(C_0, C_{16}, C_{32}, C_{48})$, each producing 4-bit output sums, (S_0, S_1, S_2, S_3) ... $(S_{60}, S_{61}, S_{62}, S_{63})$ and carry-generate/carry-propagate pairs (P_0', G_0') ... (P_{15}', G_{15}').

A second logical level of four modular four group CLA generators 250, each accepting the carry output information of a corresponding group of four CLA adders 200, generates the necessary carry information for its associated adders 200 from the four pairs of carry-generate/carry-propagate pairs and the necessary carry-generate/carry-propagate pairs, $\left[(P^*_0, G^*_0), (P^*_1, G^*_1), \text{and} (P^*_2, G^*_2)\right]$, from which the third logic level consisting of a single CLA generator 250 generates the three additional carry-in bits, (C_{16}, C_{32}, C_{48}) supplied to the first and second levels. In this manner, modular 4-bit slice CLA adders have been used to accommodate higher precision operation.

Also, the basic conditional adder unit of Figure 1 may be used as a modular adder and extended to higher precision addition by using the CLA generator concept because the logic equations defining the higher order carries are similar. For example, it may be shown (op cit Waser and Flynn) that the second level conditional same carries may be expressed as

$$C_4 = C_{N4} + C_{E4}\, C_0$$
$$C_8 = C_{N8} + C_{E8}\, C_{N4} + C_{E8}\, C_{E4}\, C_0$$
$$C_{12} = C_{N12} + C_{E12}\, C_{N8} + C_{E12}\, C_{E8}\, C_{N8} + C_{E12}\, C_{E8}\, C_{E4}\, C_0$$

Because the logic required to implement the above expressions is identical to that of the CLA generator 250 of Figure 3 and 4, a 16-bit adder may be implemented as shown in Figure 5. The adder has four conditional adders 100 connected in parallel, each accepting 4-bit pairs of operands. Each adder 100 consists of a conditional sum generator 130 and a multiplexor 150. The modular group carry-out pairs, [(C_{N4}, C_{E4}), (C_{N8}, C_{E8}), (C_{N12}, C_{E12})], are supplied to CLA generator 250 which produces the modular carry-in bits (C4, C8, C12) required to form the sixteen bit addition. The extension required to accommodate more bits clearly indicated by the CLA method previously discussed.

Because of the need for cost effective parallel fast adders, it is highly desirable that the number of processing steps required to generate the carry-bits (and hence the sum) be proportional to the logarithm of the number of bits in each operand, and at a relatively low-cost. Also, a logic structure that allows constant fan-in and fan-out and permits static versus

fixed rate pre-charge/discharge operation is desirable. The present invention is designed to achieve these goals.

SUMMARY OF THE INVENTION

A parallel N-bit binary adder network is described comprising a multiplicity of parallel modulo-2 adders, each accepting and summing corresponding operand bit pairs and a final sum carry input. The final sum carry bits are generated in parallel by a carry generating network that comprises a parallel carry propagation logic array for generating conditional carry propagation terms based on the logical OR-ing of pairs of input operand bits, an unconditional carry generation logic network based on AND-ing of pairs of input operand bits, and a logic array for operating on conditional and unconditional carry terms, in parallel for producing a set of final sum carry terms that are fed in parallel to the modulo-2 parallel adders. The number of gate delays for generating the final set of N sum carry inputs to the modulo-2 adder is $[\log_2 2N]$, providing a substantial increase in adder throughput.

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 is a logic diagram of a prior art 4-bit slice conditional sum adder.

Figure 2 is a logic diagram of a prior art 4-bit slice carry-look-ahead (CLA) adder.

Figure 3 is a logic diagram of a prior art four group CLA generator.

Figure 4 is a block diagram of a prior art 64-bit adder using full CLA.

Figure 5 is a block diagram of a prior art 16-bit conditional sum adder using a four group CLA generator.

Figure 6 is a flow diagram of a four-bit carry process.

Figure 7 is a flow diagram of an eight-bit carry process.

Figure 8 is a flow diagram of a sixteen-bit carry process.

Figure 9 is a logic diagram for a typical carry generator node implementation.

Figure 10 is a block diagram of an 8-bit carry generator.

Figure 11 is a logic diagram for an 8-bit propagation generator.

Figure 12 is a logic diagram for a one-bit adder with carry input.

Figure 13 shows a block diagram of a complete parallel adder.

Figure 14 shows the structure of an m-bit two-level carry generator module.

Figure 15 shows the interconnections for a 16-bit carry-generator using 4-bit two-level modules.

Figure 16 shows the partitioning of a 16-bit first and second level carry-generate matrix for use with 4-bit two-level modules.

Figure 17 shows the partitioning of a 16-bit third and fourth level carry-generate matrix for use with 4-bit two-level modules.

Figure 18 shows the interconnections for a 64-bit carry-generator using 8-bit two-level modules.

DETAILED DESCRIPTION OF THE INVENTION

The sum, S, of two N-bit binary number operands (A,B) where

$$A = A_{N-1}, A_{N-2}, \ldots, A_0$$

$$B = B_{N-1}, B_{N-2}, \ldots, B_0$$

may be expressed as

$$S = S_1, S_{N-2}, \ldots, S_0$$

where $S_i = A_i \oplus B_i \oplus C_{i-1}$

represents the value of the ith sum bit expressed as the modulo-2 sum of the ith operand bit values (Ai, Bi) and the carry-in bit Ci-1, from the modulo-2 sum of the next least significant bit pair (A_{i-1}, B_{i-1}). Thus, by using the boolean logic operators (•) for "AND" and (+) for "OR", the carry bits may be expressed as

$$\begin{aligned}
C_0 &= A_0 \cdot B_0 \\
C_1 &= A_1 \cdot B_1 + (A_1 + B_1) \cdot C_0 \\
C_2 &= A_2 \cdot B_2 + (A_2 + B_2) \cdot C_1 \\
&\vdots \\
C_i &= A_i \cdot B_i + (A_i + B_i) \cdot C_{i-1} \\
&\vdots \\
C_{N-1} &= A_{N-1} \cdot B_{N-1} + (A_{N-1} + B_{N-1}) \cdot C_{N-2}
\end{aligned}$$

For convenience, let

$$\begin{aligned}
G_i &= A_i \cdot B_i \\
P_i &= A_i + B_i
\end{aligned}$$

so that the above carry bit expression become

$$\begin{aligned}
C_0 &= G_0 \\
C_1 &= G_1 + P_1 C_0 \\
C_2 &= G_2 + P_2 C_1 \\
&\vdots \\
C_i &= G \\
&\vdots \\
C_{N-1} &= G_{N-1} + P_{N+1} C_{N-2}
\end{aligned}$$

(Note that for further convenience, the explicit "AND" operator symbol has been omitted so that $P_i C_i \equiv P_i \cdot C_i$). This convention will be used throughout the following description.

The above recursive expressions may be expanded as follows:

$$\begin{aligned}
C_0 &= G_0 \\
C_1 &= G_1 + P_1 G_0 \\
C_2 &= G_2 + P_2 G_1 + P_2 P_1 G_0 \\
C_3 &= G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 G_0 \\
C_4 &= G_4 + P_4 G_3 + P_4 P_3 G_2 + P_4 P_3 P_2 G_1 + P_4 P_3 P_2 P_1 G_0
\end{aligned}$$

$$C_i = G_i + P_i G_{i-1} + P_i P_{i-1} G_{i-2} + P_i P_{i-1} P_{i-2} G_{i-3} + \ldots + P_i P_{i-1} P_{i-2} \ldots P_i G_0$$

This set of equations may, in turn, be expressed in matrix form as

$$\begin{bmatrix} C_0 \\ C_1 \\ C_2 \\ C_3 \\ \vdots \\ C_i \\ \vdots \\ C_{N-1} \end{bmatrix} =$$

$$\begin{bmatrix} 1 & & & & & & & \\ P_1 & 1 & & & & & & \\ P_2P_1 & P_1 & 1 & & & & & \\ P_3P_2P_1 & P_2P_1 & P_1 & 1 & & & & \\ \vdots & \vdots & \vdots & & \ddots & & & \\ P_{N-1}P_{N-2}\cdots P_1 & P_{N-2}P_{N-3}\cdots P_1 & P_{N-3}P_{N-2}\cdots P_1 & \cdots & P_1 & 1 \end{bmatrix} \begin{bmatrix} G_0 \\ G_1 \\ G_2 \\ G_3 \\ \vdots \\ G_{N-1} \end{bmatrix}$$

or simply

$$\underline{c} = P^{(N)} \underline{g}$$

where $\underline{c}$ is the carry column vector,

$\underline{g}$ is the carry generator column vector,

and $P^{(N)}$ is the lower triangular NXN carry propagation matrix.

Thus, $\underline{g} = [G_0\ G_1\ G_2 \ldots G_{N-1}]^T = [A_0B_0\ A_1B_1\ A_2B_2 \ldots A_{N-1}B_{N-1}]^T$ represents the "AND"-ing of operand bit pairs which generate a carry-out when high. Matrix P, whose elements represent propagation control variables, describes the means by which the carry-outs are propagated to and through higher order bits.

Significantly, the P-matrix may be factorized into the product of sparse lower triangular matrices. For example,

$$P^{(3)} = \begin{bmatrix} 1 & & \\ P_1 & 1_j & \\ P_2P_1 & P_1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & & \\ 0 & 1 & \\ P_2P_1 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & & \\ P_1 & 1 & \\ 0 & P_1 & 1 \end{bmatrix}$$

$$P^{(4)} = \begin{bmatrix} 1 & & & \\ P_1 & 1 & & \\ P_2P_1 & P_1 & 1 & \\ P_3P_2P_1 & P_2P_1 & P_1 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & & & \\ 0 & 1 & & \\ 0 & 0 & 1 & \\ \prod_{1,3} & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & & & \\ 0 & 1 & & \\ \prod_{1,2} & 0 & 1 & \\ 0 & \prod_{2,3} & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & & & \\ \prod_{1,1} & 1 & & \\ 0 & \prod_{2,2} & 1 & \\ 0 & 0 & \prod_{3,3} & 1 \end{bmatrix}$$

where $\prod_{a,b} = P_a P_{a+1} P_{a+2} \cdots P_b$,

and $\prod_{a,a} = P_a$

$$P^{(7)} = \begin{bmatrix} 1 & & & & & & \\ 0 & 1 & & & & & \\ 0 & 0 & 1 & & & & \\ \prod_{1,3} & 0 & 0 & 1 & & & \\ 0 & \prod_{2,4} & 0 & 0 & 1 & & \\ 0 & 0 & \prod_{3,5} & 0 & 0 & 1 & \\ 0 & 0 & 0 & \prod_{4,6} & 0 & 0 & 1 \end{bmatrix}$$

$$\cdot \begin{bmatrix} 1 & & & & & & \\ 0 & 1 & & & & & \\ \prod_{1,2} & 0 & 1 & & & & \\ 0 & \prod_{2,3} & 0 & 1 & & & \\ 0 & 0 & \prod_{3,4} & 0 & 1 & & \\ 0 & 0 & 0 & \prod_{4,5} & 0 & 1 & \\ 0 & 0 & 0 & 0 & \prod_{5,6} & 0 & 1 \end{bmatrix}$$

$$\cdot \begin{bmatrix} 1 & & & & & & \\ \prod_{1,1} & 1 & & & & & \\ 0 & \prod_{2,2} & 1 & & & & \\ 0 & 0 & \prod_{3,3} & 1 & & & \\ 0 & 0 & 0 & \prod_{4,4} & 1 & & \\ 0 & 0 & 0 & 0 & \prod_{5,5} & 1 & \\ 0 & 0 & 0 & 0 & 0 & \prod_{6,6} & 1 \end{bmatrix}$$

$$P^{(3)} = \begin{bmatrix} 1 & & & & & & & \\ 0 & 1 & & & & & & \\ 0 & 0 & 1 & & & & & \\ 0 & 0 & 0 & 1 & & & & \\ 0 & 0 & 0 & 0 & 1 & & & \\ 0 & 0 & 0 & 0 & 0 & 1 & & \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & \\ \prod_{4,7} & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\cdot \begin{bmatrix} 1 & & & & & & & \\ 0 & 1 & & & & & & \\ 0 & 0 & 1 & & & & & \\ \prod_{1,3} & 0 & 0 & 1 & & & & \\ 0 & \prod_{2,4} & 0 & 0 & 1 & & & \\ 0 & 0 & \prod_{3,5} & 0 & 0 & 1 & & \\ 0 & 0 & 0 & \prod_{4,6} & 0 & 0 & 1 & \\ 0 & 0 & 0 & 0 & \prod_{5,7} & 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & & & & & & & \\ 0 & 1 & & & & & & \\ \prod_{1,2} & 0 & 1 & & & & & \\ 0 & \prod_{2,3} & 0 & 1 & & & & \\ 0 & 0 & \prod_{3,4} & 0 & 1 & & & \\ 0 & 0 & 0 & \prod_{4,5} & 0 & 1 & & \\ 0 & 0 & 0 & 0 & \prod_{5,6} & 0 & 1 & \\ 0 & 0 & 0 & 0 & 0 & \prod_{6,7} & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & & & & & & & \\ \prod_{1,1} & 1 & & & & & & \\ 0 & \prod_{2,2} & 1 & & & & & \\ 0 & 0 & \prod_{3,3} & 1 & & & & \\ 0 & 0 & 0 & \prod_{4,4} & 1 & & & \\ 0 & 0 & 0 & 0 & \prod_{5,5} & 1 & & \\ 0 & 0 & 0 & 0 & 0 & \prod_{6,6} & 1 & \\ 0 & 0 & 0 & 0 & 0 & 0 & \prod_{7,7} & 1 \end{bmatrix}$$

Thus, at each binary increment, $2^k \leq r < 2^{k+1}$, $P^{(r)}$ is factorizable into (k+1) lower triangle matrices of the form shown. These factorized equations may be represented by the flow diagrams of Figure 6, 7 and 8.

Figure 6 corresponds to the four-bit carry propagation process represented by the factorization of $P^{(4)}$. The input to the process consists of the carry-generator vector, $[G_0\ G_1\ G_2\ G_3]^T$ shown at the bottom. The diagonal lines with arrow-heads correspond to multiple ("AND") operations on the data of the node of origin by the corresponding labelled expression. Unlabeled vertical lines between nodes represent transmission paths with no modification of data transmitted from a lower node to a higher node. All nodes are summing ("OR") junctions. For example, $C_1 = G_1 + P_1G_0$ and $C_3 = P_3P_2\ (G_1 + P_1G_0)$ $+ (G_3 + P_3G_2) = P_3P_2P_1G_0 + P_3P_2G_1 + P_3G_2 + G_3$.

The carry-out vector, $[C_0\ C_1\ C_2\ C_3]^T$, is represented by the values present at the upper output nodes.

Figure 7 and 8 show flow diagrams for $P^{(8)}$ and $P^{(16)}$, respectively representing 8 and 16 bit carry generation processes. Clearly, flow diagrams for greater number of bits may be generated in a similar fashion by extending the principles expounded.

For each binary increment, $2^k \leq r \leq 2^{k+1} - 1$, or for each doubling of the number of bits used in the operands, one additional sparse lower triangular matrix is required to represent the factorized form of the $P^{(r)}$ matrix. Thus, for $2 \leq r \leq 3$, $P^{(r)}$ factors into 2 matrices; for $4 \leq r \leq 7$, $P^{(r)}$ factors into 3 matrices, and for $2^k \leq r \leq 2^{k+1}-1$, $P^{(r)}$ factors into (k+1) matrices.

Each factorized matrix operation corresponds to a row of nodes shown in Figures 6, 7, and 8. The lowest (zero) level nodes correspond to the input carry generate vector values, $\underline{g}$. The values at the next level of nodes corresponds to the column vector that would obtain if the extreme right hand factorized matrix of the examples given above were to operate on the input generate vector, $\underline{g}$. Similarly, the second level of nodes has values corresponding to that which would obtain if the second most extreme right had factorized matrices operated on the vector resulting from product to its right. And so on for succeeding levels.

In general, k + 1 factorized matrices (stages) are required for 2^{k+1} bits in each operand, i.e., $[\log_2 N]$ stages for N-bit operands.

The flow diagrams of Figures 6, 7 and 8 also imply the logic network structures shown in Figures 9 through 11.

Figure 9 represents a typical nodal processor 10 located at, say, node l, k of Figure 8 producing $G_{l,k}$ at its output. Processor 10 accepts as input operands $G_{l-1,\ k-2^{l-1}}$, $G_{l-1,\ k}$ and $P_k P_{k-1} \cdots P_{k-2^{l-1}}$ at its input terminals 11, 12 and 13 respectively. "AND"-gate 16 and "OR"-gate operate on these inputs to produce at output 14 the boolean function

$$G_{l,k} = G_{l-1,\ k} + P_k P_{k-1} \cdots P_{k-2^{l-1}} G_{l-1,\ k-2^{l-1}}$$

Figure 10 is an embodiment of an 8-bit carry generator having four rows (0-3) and 8 columns (0-7). Rows 1 through 3 comprises 7, 6 and 4 nodal processors 10, respectively, each of the type depicted in Figure 9. Row 0 comprises 8 AND-gates 20 arranged to accept at input terminals 301 corresponding operand bit pairs, $\{A_k, B_k\}$, forming $G_{0,k} = A_k \cdot B_k$ and supplied to processors 10 on line 11. The processors 10 of row 1 also accept the seven propagation variable P_1 through P_7 on input lines 305. Propagation variable P_k being applied as an input to processor 10 located at row 1, column k on line 13 together with $G_{0,k-1}$ supplied by lines 12. The output of processor 10 located at 1,k is

$$G_{1,k} = G_{0,k} + P_k G_{0,k-1}$$

In a similar manner, processors 10 of row 2 are supplied the outputs of row 1 together with propagation variable P_{21} through P_{76} from input line 307. The output of processor 10 located at 2, k is

$$G_{2,k} = G_{1,k} + P_k P_{k-1} G_{1,k-2}$$

Processor 10 at location 3, k in a similar manner generates an output

$$G_{3,k} = G_{2,k} + P_k P_{k-1} P_{k-2} P_{k-3} G_{2,k-4}$$

from inputs provided by lower level processors and propagation variable $P_4P_3P_2P_1$ through $P_7P_6P_5P_4$ supplied on input lines 309.

Carry output C_0 is available directly from AND-gate 20 at location 0,0 on line 303; C_1 from output line 14 of processor 10 at location 1,1; C_2 and C_3 from processors 10 at location 2,2, and 2,3 respectively; and C_4 through C_7 from row 3 processor 10 outputs.

It is clear, by reference to the flow diagrams of Figures 6, 7 and 8 and carry generators 300 of Figure 10, that the architecture and organization of the 8-bit carry generator 300 may be expanded indefinitely adding an additional row each time the number of bits in each operand is doubled. The number of parallel processors required in each row is summarized in Table I.

		Operand Bits				
		4	8	16	32	64
Row	0	4	8	16	32	64
	1	3	7	15	31	63
	2	2	6	14	30	62
	3		4	12	28	60
	4			8	24	56
	5				16	48
	6					32

Table I

Figure 11 is a logic circuit for implementing an 8-bit propagation generator suitable for supplying propagation variables to the 8-bit carry generator of Figure 10. Propagation generator 400 comprises 7 OR-gates 40 in row 0 used to form propagation variables $P_1, P_2, ..., P_7$ from input operand bit pairs $\{A_k, B_k\}$ as follows:

$$P_k = A_k + B_k$$

The set, $\{P_k\}$, is available on output lines 307. Subsequent rows are comprised of AND-gates 50. The k^{th} AND-gate of row 1 accepts the k^{th} and $k\text{-}1^{th}$ output of row 0 to form at its output 307 $P_k P_{k-1}$. Similarly, the kth processor of row 2 accepts the k^{th} and $k\text{-}2^{th}$ output of row 1 to form the set of propagation variables, $\{P_k P_{k-1} P_{k-2} P_{k-3}\}$, provided at output 309.

Clearly, the organization and architecture of processor 400 may be extended to accommodate more operand bits by extending the structure of Figure 11 to the left and adding an additional row of AND-gates 50 each time the number of input operand bits are doubled. The number of gates required per row are indicated in Table II.

Operand Bits

Row	4	8	16	32	64
0	3	7	15	31	63
1	2	6	14	30	62
2		4	12	28	60
3			8	24	56
4				16	48
5					32

Table II

Figure 12 represents a logic network 60 for forming the complete bit sum of two operand bits (A_k, B_k) and a carry-in bit Ck comprising exclusive-or (EOR) networks 61 and 62. EOR network 61 forms the modulo-2 sum $A_k \oplus B_k$ and network 62 produces at its output

$$S_k = A_k \oplus B_k \oplus C_{k-1}$$

Based on the preceding descriptions summer network 60, carry generator 300 and propagation generator 400, a complete parallel binary adder may be defined as shown in Figure 13, organized to accept two N-bit operands

$$A = A_0\, A_1\, A_2 \ldots A_{N-1}$$

$$B = B_0\, B_1\, B_2 \ldots B_{N-1}$$

Operands A and B are applied to the inputs of propagation generator 400, carry generator 300 and sum unit 500. Propagation generator 400 and carry generator 300 are configured in accordance with the prior description. Sum unit 500 comprises N one-bit plus carry-in bit EOR networks 60, each as described in Figure 12. The carry-in to each EOR network 60 is provided by the appropriate output terminal of carry generator 300. Propagation variables are provided to carry generator 300 by propagation generator 400 as determined by the two input operands A and B. The output of sum unit 500 is

$$S = S_0\, S_1 \ldots S_{N-1}$$

where

$$S_k = A_k \oplus B_k \oplus C_{k-1}$$

Note that carry C_{N-1} is available at the output as an overflow bit of for use in extending the number of bits in the operands A and B.

The preferred implementation of carry generation 300 uses modular medium scale integrated circuit technology. For example, by properly sub-sectioning the flow graph of Figure 8 into seven subsections as shown by the dotted outlines, a 4-bit wide and 2-level deep module may be defined that forms the basis for a modular building-block approach to the circuit implementation. The 4-bit wide partitioning is somewhat arbitrary and is mainly chosen for purposes of explanation because it probably represents the lowest level of modularization that allows the principle of modularity to be described.

Figure 14 is a block diagram of an m-bit wide, 2-level module 500 comprising two layers of m nodal processors 10 of the type shown in Figure 9. Five sets of m-input lines are accommodated: inputs 501 accept the corresponding l-level outputs, $\{G_{l,k}\}$; inputs 503 accept the l-level outputs displaced by 2^{l-1}, $\{G_{l,k-2^{l-1}}\}$; inputs 505 and 507 accept the conditional carry-terms $\left\{\prod_{k,k+2^{l+1}-l}\right\}$ and $\left\{\prod_{k,k+2^{l}-1}\right\}$, respectively; and inputs 509 accept the $(l+1)^{th}$ inner layer output terms (displaced by 2^l), $\{G_{l+1,k-2^l}\}$.

Two sets of output lines are provided: outputs 511 correspond to the first layer output terms, $\{G_{l+1,k}\}$; and outputs 513 are the second layer (or module) outputs, $\{G_{l+2,k}\}$.

Figure 15 is an interconnection diagram for a carry generator 300 using 4-bit wide (m=4) 2-layer modules 500. Each logic unit 520 represents a set of four unit 20 AND-gates used to form $\{G_k\}$.

Figure 15 may be best understood by referring to Figure 16 that shows the matrix equation relating the zero level (l=0) inputs, $\{G_{0,k}\}$, to the second level (l=2) outputs, $\{G_{2,k}\}$, and to Figure 17 showing the matrix equation relating the second level outputs to the fourth level outputs, $\{G_{4,k}\}$. In Figure 16, the two 16x16 matrices $\left(P_1^{(16)}, P_2^{(16)}\right)$ are each partitioned into 16 4x4 submatrices. Each non-zero valued submatrix corresponds to a

single layer 4-bit wide operation performed within a 500 module. The submatrices of the right-hand matrix correspond to first layer operations while those in the left-hand matrix correspond to the second layer operations previously described. Similarly, the right hand set of submatrices in Figure 17 corresponds to third level (l=3) operations and the left set corresponds to fourth level (l=4) operations. These equations provide interconnect information by relating the individual module 500 inputs to their outputs.

For example, consider the input/output relationship of module 500 in the first row of Figure 15 identified by coordinates (1,3).

$$G_{2,8\text{-}11}=P_{2,32}\cdot P_{1,21}G_{0,0\text{-}3}+P_{2,32}\cdot P_{1,22}\cdot G_{0,4\text{-}7}+P_{2,33}P_{1,32}G_{0,4\text{-}7}+P_{2,33}G_{0,8\text{-}11}$$

Because $P_{2,32}\cdot P_{1,21}=0$

$$G_{2,8\text{-}11}=P_{2,32}\cdot P_{1,22}G_{0,4\text{-}7}+P_{2,33}P_{1,32}G_{0,4\text{-}7}+P_{2,33}G_{0,8\text{-}11}$$

$$=\begin{bmatrix}0 & 0 & \prod_{7,8} & 0\\ 0 & 0 & 0 & \prod_{8,9}\\ 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0\end{bmatrix}\begin{bmatrix}1 & & & \\ \prod_{5,15} & 1 & & \\ 0 & \prod_{6,6} & 1 & \\ 0 & 0 & \prod_{7,7} & 1\end{bmatrix}\begin{bmatrix}G_{0,4}\\ G_{0,5}\\ G_{0,6}\\ G_{0,7}\end{bmatrix}$$

$$+\begin{bmatrix}1 & & & \\ 0 & 1 & & \\ \prod_{9,10} & 0 & 1 & \\ 0 & \prod_{10,11} & 0 & 1\end{bmatrix}\begin{bmatrix}0 & 0 & 0 & \prod_{8,8}\\ 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0\end{bmatrix}\begin{bmatrix}G_{0,4}\\ G_{0,5}\\ G_{0,6}\\ G_{0,7}\end{bmatrix}$$

$$+\begin{bmatrix}1 & & & \\ 0 & 1 & & \\ \prod_{9,10} & 0 & 1 & \\ 0 & \prod_{10,11} & 0 & 1\end{bmatrix}\begin{bmatrix}1 & & & \\ \prod_{9,9} & 1 & & \\ 0 & \prod_{10,10} & 1 & \\ 0 & 0 & \prod_{11,11} & 1\end{bmatrix}\begin{bmatrix}G_{0,8}\\ G_{0,9}\\ G_{0,10}\\ G_{0,11}\end{bmatrix}$$

Because $$\begin{bmatrix}G_{0,4}\\ G_{1,5}\\ G_{1,6}\\ G_{1,7}\end{bmatrix}=\begin{bmatrix}1 & & & \\ \prod_{5,5} & 1 & & \\ 0 & \prod_{6,6} & 1 & \\ 0 & 0 & \prod_{7,7} & 1\end{bmatrix}\begin{bmatrix}G_{0,4}\\ G_{0,5}\\ G_{0,6}\\ G_{0,7}\end{bmatrix}$$

and $$\begin{bmatrix} G_{0,8} \\ G_{1,9} \\ G_{1,10} \\ G_{1,11} \end{bmatrix} = \begin{bmatrix} 1 & & & \\ \prod_{9,9} & 1 & & \\ 0 & \prod_{10,10} & 1 & \\ 0 & 0 & \prod_{11,11} & 1 \end{bmatrix} \begin{bmatrix} G_{0,8} \\ G_{0,9} \\ G_{0,10} \\ G_{0,11} \end{bmatrix}$$

$$G_{2,8-11} = \begin{bmatrix} 0 & 0 & \prod_{7,8} & 0 \\ 0 & 0 & 0 & \prod_{8,9} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} G_{0,4} \\ G_{1,5} \\ G_{1,6} \\ G_{1,7} \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & \prod_{8,8} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \prod_{8,8} \prod_{9,10} \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} G_{0,4} \\ G_{0,5} \\ G_{0,6} \\ G_{0,7} \end{bmatrix}$$

$$+ \begin{bmatrix} 1 & & & \\ 0 & 1 & & \\ \prod_{9,10} & 0 & 1 & \\ 0 & \prod_{10,11} & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & & & \\ \prod_{9,9} & 1 & & \\ 0 & \prod_{10,10} & 1 & \\ 0 & 0 & \prod_{11,11} & 1 \end{bmatrix} \begin{bmatrix} G_{0,8} \\ G_{0,9} \\ G_{0,10} \\ G_{0,11} \end{bmatrix}$$

This latter equation expresses the required inputs to module 500 (1,3): the first expression on the right implies only two non-zero products $\prod_{7,8} \cdot G_{1,6}$ and $\prod_{8,9} \cdot G_{1,7}$ thus requiring $\prod_{7,8}$, $\prod_{8,9}$, $G_{1,6}$ and $G_{1,7}$ as inputs; the second expression requires $\prod_{8,8}$, $\prod_{9,10}$; and the third requires the input quadruplet $[G_{0,8}\ G_{0,9}\ G_{0,10}\ G_{0,11}]^T$, and the triplet $\left[\prod_{9,9}, \prod_{10,10}, \prod_{11,11}\right]$. Summarizing, the required inputs are: $G_{0,8\text{-}11}$, $G_{1,6\text{-}7}$, $G_{0,7}$, and $\prod_{9-11,9-11}$, as shown in Figure 15. (Please note that for Figure 15, the output carries, $\{C_k\}$, are equal to $\{G_{4,k}\}$.)

A similar analysis for module 500 (2,4) results in the following expression:

$$\begin{bmatrix} C_{12} \\ C_{13} \\ C_{14} \\ C_{15} \end{bmatrix} = \begin{bmatrix} G_{4,12} \\ G_{4,13} \\ G_{4,14} \\ G_{4,15} \end{bmatrix} = \begin{bmatrix} \prod_{5,12} & G_{3,4} \\ \prod_{6,13} & G_{3,5} \\ \prod_{7,14} & G_{3,6} \\ \prod_{8,15} & G_{3,7} \end{bmatrix} + \begin{bmatrix} \prod_{9,12} & G_{2,8} \\ \prod_{10,13} & G_{2,9} \\ \prod_{11,14} & G_{2,10} \\ \prod_{12,15} & G_{2,11} \end{bmatrix} + \begin{bmatrix} G_{2,12} \\ G_{2,13} \\ G_{2,14} \\ G_{2,15} \end{bmatrix}$$

The interconnections shown in Figure 15 for module 500 (2,4) result.

Figure 18 shows a simplified interconnection diagram for a 64-bit carry generator using 3-layers of 8-bit wide two layer modules. Specific details of the interconnections may be obtained by partitioning the carry-generator 300 matrices in the same manner as shown for the 4-bit wide two layer example. For the 64-bit case, however, three sets of equations, corresponding to the three layers of Figure 18, must be used.

Another preferred embodiment using a slightly different concept of modularity is shown in Figure 19. For purposes of explanation, a 24-bit adder network is shown comprising: three 8-bit conditional adder networks 141 each accepting two eight bit operands [(A_{0-7}, B_{0-7}), (A_{8-15}, B_{8-15}), (A_{16-23}, B_{16-23})]. and each outputting two conditional 8-bit sums (S_E,S_N) as previously described in Figure 1; multiplexer units 160 for selecting the S_E or S_N output of each conditional adder unit which is controlled by a two state carry signal; carry and propagation generator units 600 each comprising a carry generator 400 for accepting two 8-bit operands and producing at its output the highest carry, say, out of a possible set of (C_0, C_1, ..., C_7) for controlling its associated 2:1 MUX 160. Note that the lowest order (extreme left) MUX 160 is shown dotted so as to indicate that modularity consideration may require that each 8-bit conditional adder 141 be packaged with an associated MUX 160, in which case its control but would be set low because the absence of an input carry makes the S_N output always valid. In effect, each of the three vertical grouping of units 141, 160 and 600 constitute a modular adder and carry-out generator 700 requiring its associated two fields of operand bits and carry-in bit. The tandem ensemble of these units makes-up the complete adder. The output sum is represented by the 25-bit sum S_{0-7}, S_{8-15}, S_{16-23}, S_{24}.

In order to accommodate the carry-in bits (C-1, C7, C15) to units 600, a slight modification of the basic matrix and flow diagram must be made. Consider, the unit 600 shown on the extreme right of Figure 19. The requisite matrix has the form

$$\begin{bmatrix} C_{-1} \\ C_0 \\ C_1 \\ C_2 \\ C_3 \\ \vdots \\ C_7 \end{bmatrix} =$$

$$\begin{bmatrix} 1 & & & & & & & \\ P_0 & 1 & & & & & & \\ P_0P_1 & P_1 & 1 & & & & & \\ P_0P_1P_2 & P_1P_2 & P_2 & 1 & & & & \\ P_0P_1P_2P_3 & P_1P_2P_3 & P_2P_3 & P_3 & 1 & & & \\ \vdots & \vdots & \vdots & \vdots & & \ddots & & \\ P_0P_1\cdots P_7 & P_1P_2\cdots P_7 & P_2P_3\cdots P_7 & P_3P_4\cdots P_7 & P_4P_5\cdots P_7 & \cdots & 1 \end{bmatrix} \begin{bmatrix} C_{-1} \\ G_0 \\ G_1 \\ G_2 \\ G_3 \\ \vdots \\ G_7 \end{bmatrix}$$

Note that if the carry-in, C_{-1}, is zero (non-existent), the first row and column are zero. Also, $P_0=C_{-1}$ so that P0 and all its product terms vanish if $C_{-1}=0$. Thus, when $C_{-1}=0$, networks 300 and 400 are as previously defined. If $C_1=1$ when the form of the matrix, carry-generator network 300 and propagation generator 400 have the same logic structure as previously described.

For example, Figure 20 shows the flow diagram corresponding to a 4-bit carry-generator 300 with an input carry bit C_{-1}, suitable for concatenating 4-bit conditional adder units in a similar fashion to that shown for 8-bit conditional adder units 141 in Figure 19. The necessary steps required for generating the output carry, C_3, are shown by solid lines while the dotted-lines represent the other possible, but not required, processing steps previously shown. This implies the carry-out generator structure 300' shown in Figure 21 using processing modules 10.

Figure 22 and Figure 23 are the corresponding flow diagram and simplified carry-out generator structure 300 for an 8-bit unit respectively, as used in the adder network of Figure 19.

Figure 24 shows a simplified block diagram for propagation generator 400' suitable for use with the 8-bit adder module 700 of Figure 19. The subset of propagation terms required the 4-bit carry-out generator 300' ($P_3P_2P_1P_0$, P_3P_2, P_3, P_1) is also available from this unit.

The same flow diagram and logic networks are applicable to all concatenated units 600 of Figure 19. However, in the case of the lowest significant unit 600 at the extreme right of Figure 19, the carry-in from the previous stage is non-existent so that $C_{-1}=0$. For the other stages, the carry-out of the previous section is used as the carry-in.

Clearly, the concept of modular carry propagation for extended operand precision, an example of which is shown in Figure 19 is adaptable to the use of 4, 8, 16, ... or any other size modular bit units by implementing units 600, 160 and 141 for the word size desired. Also, mixed systems in which associated units 600, 160, and 141 of a given 700 section, are of the same word size, but not necessarily the same word size the other 700 units tandemly connected with it, can be constructed.

These and other similar variations will become apparent to those versed in the art.

CLAIMS

1. A parallel carry generating network for use in a multibit binary adder comprising:

(a) means for accepting a first and second N-bit binary operands;

(b) carry propagation logic network array for generating conditional carry propagation terms by parallel logical operations on pairs of corresponding bits of said first and second operands; and

(c) carry generator logic array for generating final sum carry bits, comprising:

i) logic network for generating unconditional carry terms by parallel operation on said pairs of corresponding bits of said first and second operands; and

ii) logic network array for accepting and operating on said unconditional carry terms and said conditional carry terms for producing a parallel set of final sum carry terms.

2. A parallel carry generating network as in Claim 1 wherein said carry propagation logic further comprises:

(a) a zero level (l=0) multiplicity of N-1 gates for the logical OR-ing of said corresponding bit pairs for all said bit pairs except the least significant, producing at each gate output, its associated first level conditional carry term arranged in ascending order of said operand bit pair inputs;

(b) a first level (l=1) multiplicity of N-2 AND-gates, each said gate accepting as inputs adjacent overlapping ascending order outputs producing an ordered ascending set of second level conditional carry terms; and

(c) additional higher levels of AND-gates, each succeeding level having $N-2^l$ AND-gates, each said gates having a first input connected its corresponding lower level ordered output and second input connected to the 2^lth lesser ordered output of said lower level, for forming the required higher level up to and including level, l=L, for which $2^l = N/2$; each level producing

succeeding higher order conditional carry terms at the individual AND-gate outputs.

3. A parallel carry generating network as in Claim 1 wherein said carry generator logic array further comprises:

(a) a zero level (l=0) multiplicity of N gates in parallel for the logical AND-ing of said corresponding operand bit pairs producing at each gate output the unconditional carry term associated with said operand input bit pair, said outputs being arranged in ascending order corresponding to the ascending order of said operand input bit pairs;

(b) carry generator processor having a first , second and third input, said first and second inputs connected to distinct partial carry terms of the same next lower level and said third input connected to an associated conditional carry term provided by said conditional carry generator for producing at its output a next level partial carry term;

(c) a first level (l=1) multiplicity of N-1 processors arranged in ascending order each having a first, second and third input, each said first input connected to one corresponding zero level processor outputs for all outputs except that associated with the least significant of said input bit pairs, said second input of each processor connected to the output of the zero level gate associated with the next least significant of said input bit pairs, each said third processor input connected to the corresponding zero level output of said carry propagation logic, producing at a each processor output a partial carry term of second order arranged in ascending order; and

(d) additional higher levels (l = 2, 3, ..., L) of processors each level having $N-2^{l-1}$ processors, each said first input connected to one said corresponding next lower level processor output for all outputs in ascending order beginning at the 2^{l-1} output, said second input of each processor connected to the output of the next lower level processor at 2^{l-1} positions below said corresponding next lower level processor input to said first input, and said third input connected to the corresponding l^{th} level conditional carry generator output.

4. A parallel carry generator network as in Claim 3 wherein said carry processor comprises:

(a) OR-gate with an output and a first and second input ; said first input connected to said processor first input, said OR-gate output being said processor output; and

(b) AND-gate with its output connected to said OR-gate second input, with a first and second input, said first input connected to said processor second input and said second AND-gate input connected to said processor third input.

5. A parallel adder network comprising:

(a) parallel sum unit having first, second and third set of inputs, said first and second input set for accepting a first and second input operand, said third input set connected to the output of a parallel carry generating network, for accepting parallel carry bits and producing the sum of the input operands.

(b) parallel carry generating network having a first and second operand and generating at its output terminals a parallel set of carry bits.

6. A parallel adder network as in Claim 5 wherein said parallel sum unit further comprises a multiplicity of sum units, one for each operand bit pair comprising adder means for forming the modulo-2 sum of each corresponding operand bit pair and carry bit.

7. A parallel adder network as in Claim 6 wherein said adder means comprises exclusive-or gates.

8. A carry generating module comprising:

(a) an equal number of first and second level carry generator processors each having a first, second and third input, said first and second input connected to distinct partial carry terms from the corresponding next lower level output, said second input connected to associated displaced partial carry terms from the next lower layer output and said third input

connected to an associated conditional carry term from the next lower layer output, for producing at each said carry generator processor output a next level partial carry term;

(b) means for connecting said output of each said first level carry generator processor to a corresponding said second level processor first input;

(c) means for connecting said first and second level processor outputs to a first set of external terminals;

(d) means for connecting said first and second level processor second and third inputs to a second set of external terminals; and

(e) means for supporting said network as a unitized structure.

9. A multibit adder network comprising:

(a) modular adder and carry-out generator unit capable of accepting two fields of operand bits and an input carry bit for producing the sum and output carry, said modular adder and carry-out generator comprising a conditional sum adder, conditional sum selector means controlled by the input carry, and a parallel carry generator unit for generating said output carry from said operand bits and input carry bits;

(b) multiplicity of said modular adder and carry-out generator units connected in tandem so that the prior modular adder and carry-out generator carry-out is connected to the control input of said conditional sum based on a carry-in is selected when said prior carry-out is asserted, otherwise selecting the other conditional sum, said selected sums and final carry-out being representative of the desired adder output.

10. A modular multibit adder network as in claim 9 wherein each said modular adder and carry-out generator units accepts two input operand pairs comprising a like number of bits.

11. A modular multibit adder network as in claim 9 wherein each said modular adder and carry-out generator units may each accept input operand pairs of differing numbers of bits.

12. A modular carry propagation unit for use in a modular multibit adder network comprising:

(a) a simplified carry generator for generating an output carry bit from the two sets of operand bits and a carry-in bit comprising a minimum set of processor elements necessary to generate said output carry bit, controlled by an associated minimum set of propagation variables; and

(b) a simplified propagation generator for generating propagation control variables from the two sets of operand bits and a carry-in bit comprising a minimum set of logic elements necessary to generate said minimum set of propagation variables needed to control said simplified carry generator.